

**PCI-9113**  
**32 Channels Isolation**  
**Analog Input Card**



@Copyright 1997~1998 ADLink Technology Inc.  
All Rights Reserved.

Manual Rev 2.1: September 5, 1998

The information in this document is subject to change without prior notice in order to improve reliability, design and function and does not represent a commitment on the part of the manufacturer.

In no event will the manufacturer be liable for direct, indirect, special, incidental, or consequential damages arising out of the use or inability to use the product or documentation, even if advised of the possibility of such damages.

This document contains proprietary information protected by copyright. All rights are reserved. No part of this manual may be reproduced by any mechanical, electronic, or other means in any form without prior written permission of the manufacturer.

### **Trademarks**

PCI-9113 is a registered trademark of ADLink Technology Inc., IBM PC is a registered trademark of International Business Machines Corporation. Intel is a registered trademark of Intel Corporation. Borland C++ is a registered trademark of Borland International, Inc. Other product names mentioned herein are used for identification purposes only and may be trademarks and/or registered trademarks of their respective companies.



# Contents

How to Use This Manual .....	vii
------------------------------	-----

Introduction .....	1
--------------------	---

1.1 Software Supporting.....	2
1.2 Features.....	2
1.3 Applications .....	3
1.4 Specifications .....	3

Installation.....	7
-------------------	---

2.1 What You Have.....	7
2.2 Unpacking .....	8
2.3 Device Installation for Windows 95.....	8
2.4 PCI-9113's Layout .....	11
2.5 Jumper Descriptions .....	12
2.5.1 Polarity Selection Jumper.....	12
2.5.2 Full Range Jumper .....	12
2.5.3 Possible AD Input Range Configurations .....	12
2.6 PCI Configuration .....	13

Signal Connections .....	15
--------------------------	----

3.1 Connectors Pin Assignment .....	15
-------------------------------------	----

<b>3.2</b>	<b>Analog Input Signal Connection.....</b>	<b>16</b>
<b>3.3</b>	<b>Daughter Board Connection .....</b>	<b>17</b>
3.3.1	<i>Connect with ACLD-9881 .....</i>	<i>17</i>
3.3.2	<i>Connect with ACLD-9137 .....</i>	<i>17</i>
3.3.3	<i>Connect with ACLD-9188.....</i>	<i>17</i>

## Registers Structure & Format ..... 19

<b>4.1</b>	<b>I/O Port Address.....</b>	<b>19</b>
<b>4.2</b>	<b>A/D Data Registers .....</b>	<b>20</b>
<b>4.3</b>	<b>A/D Channel Control Register .....</b>	<b>21</b>
<b>4.4</b>	<b>A/D Input Signal Range Control Register .....</b>	<b>21</b>
<b>4.5</b>	<b>A/D Status Readback Register .....</b>	<b>22</b>
<b>4.6</b>	<b>A/D Trigger Mode Control and Readback Register .....</b>	<b>22</b>
<b>4.7</b>	<b>Software Trigger Register .....</b>	<b>23</b>
<b>4.8</b>	<b>Interrupt Control and Readback Register.....</b>	<b>23</b>
<b>4.9</b>	<b>Hardware Interrupt Clear Register .....</b>	<b>24</b>
<b>4.10</b>	<b>A/D Data and Channel Number Registers.....</b>	<b>24</b>
<b>4.11</b>	<b>High Level Programming.....</b>	<b>25</b>
<b>4.12</b>	<b>Low-Level Programming .....</b>	<b>25</b>

## Operation Theorem ..... 27

<b>5.1</b>	<b>A/D Conversion.....</b>	<b>27</b>
5.1.1	<i>A/D Conversion Procedure .....</i>	<i>28</i>
5.1.2	<i>A/D Signal Source Control.....</i>	<i>28</i>
5.1.3	<i>A/D Trigger Source Control.....</i>	<i>30</i>
5.1.4	<i>A/D Data Transfer Modes.....</i>	<i>31</i>

5.1.5	<i>A/D Data Format</i> .....	34
<b>5.2</b>	<b>Interrupt Control</b> .....	<b>34</b>
5.2.1	<i>System Architecture</i> .....	34
5.2.2	<i>IRQ Level Setting</i> .....	35
5.2.3	<i>Dual Interrupt System</i> .....	35
5.2.4	<i>Interrupt Source Control</i> .....	36
<b>5.3</b>	<b>Timer/Counter Operation</b> .....	<b>36</b>
5.3.1	<i>Introduction</i> .....	37
5.3.2	<i>Pacer Trigger Source</i> .....	37

## C/C++ Software Library ..... 39

<b>6.1</b>	<b>Installation</b> .....	<b>39</b>
6.1.1	<i>Installation</i> .....	39
<b>6.2</b>	<b>C/C++ Programming Library</b> .....	<b>41</b>
6.2.1	<i>Data Types</i> .....	41
6.2.2	<i>_9113_Initial</i> .....	42
6.2.3	<i>_9113_Software_Reset</i> .....	42
6.2.4	<i>_9113_AD_Read_Data</i> .....	43
6.2.5	<i>_9113_AD_Read_Data_Repeat</i> .....	44
6.2.6	<i>_9113_AD_Read_Data_MUX</i> .....	45
6.2.7	<i>_9113_AD_Read_Data_Repeat_MUX</i> .....	45
6.2.8	<i>_9113_AD_Set_Channel</i> .....	46
6.2.9	<i>_9113_AD_Set_Range</i> .....	47
6.2.10	<i>_9113_AD_Get_Range</i> .....	48
6.2.11	<i>_9113_AD_Get_Status</i> .....	50
6.2.12	<i>_9113_AD_Set_Mode</i> .....	51
6.2.13	<i>_9113_AD_Get_Mode</i> .....	51
6.2.14	<i>_9113_INT_Set_Reg</i> .....	52
6.2.15	<i>_9113_AD_Get_Reg</i> .....	53

6.2.16	<i>_9113_Reset_FIFO</i> .....	53
6.2.17	<i>_9113_AD_Soft_Trigger</i> .....	54
6.2.18	<i>_9113_Set_8254</i> .....	55
6.2.19	<i>_9113_Get_8254</i> .....	56
6.2.20	<i>_9113_AD_Timer</i> .....	56
6.2.21	<i>_9113_Counter_Start</i> .....	57
6.2.22	<i>_9113_Counter_Read</i> .....	58
6.2.23	<i>_9113_Counter_Stop</i> .....	59
6.2.24	<i>_9113_INT_Source_Control</i> .....	59
6.2.25	<i>_9113_CLR_IRQ</i> .....	60
6.2.26	<i>_9113_Get_IRQ_Channel</i> .....	61
6.2.27	<i>_9113_Get_IRQ_Status</i> .....	62
6.2.28	<i>_9113_AD_FFHF_Polling</i> .....	62
6.2.29	<i>_9113_AD_FFHF_Polling_MUX</i> .....	63
6.2.30	<i>_9113_AD_Aquire</i> .....	64
6.2.31	<i>_9113_AD_Aquire_MUX</i> .....	65
6.2.32	<i>_9113_AD_INT_Start</i> .....	66
6.2.33	<i>_9113_AD_FFHF_INT_Start</i> .....	68
6.2.34	<i>_9113_AD_INT_Status</i> .....	70
6.2.35	<i>_9113_AD_FFHF_INT_Status</i> .....	71
6.2.36	<i>_9113_AD_FFHF_INT_Restart</i> .....	71
6.2.37	<i>_9113_AD_INT_Stop</i> .....	73

## Calibration & Utilities ..... 75

<b>7.1</b>	<b>What do you need</b> .....	<b>75</b>
<b>7.2</b>	<b>VR Assignment</b> .....	<b>76</b>
<b>7.3</b>	<b>A/D Adjustment</b> .....	<b>76</b>
7.3.1	<i>Uni-polar input</i> .....	76
7.3.2	<i>Bi-polar input</i> .....	76



7.4	Software A/D Offset Calibration .....	77
	Software Utility .....	79
8.1	9113util .....	79
8.1.1	Running 9113util.exe .....	79
8.1.2	System Configuration.....	80
8.1.3	Calibration .....	81
8.1.4	Functional Testing .....	83
8.2	I_EEPROM .....	84
8.2.1	Running I_eeeprom.exe .....	84
	Product Warranty/Service .....	87



# How to Use This Manual

This manual is designed to help you to use the PCI-9113. The manual describes the versatile functions and the operation theorem of the PCI-9113 card. It is divided into six chapters:

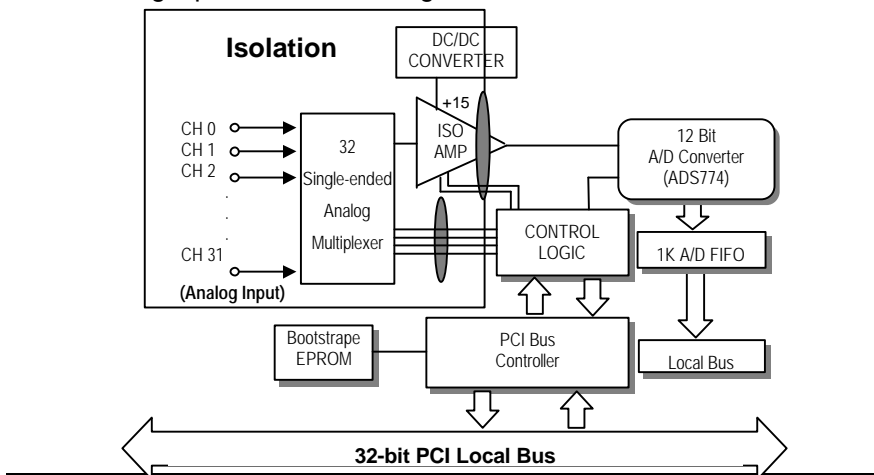
- **Chapter 1**, "Introduction," gives an overview of the product features, applications, and specifications.
- **Chapter 2**, "Installation," describes how to install the PCI-9113. The layout of PCI-9113 is shown, jumper setting for analog input channel configuration.
- **Chapter 3**, "Signal Connection," describes the connectors' pin assignment and how to connect the outside signal and devices with the PCI-9113.
- **Chapter 4**, "Registers Structure & Format," describes the details of register format and structure of the PCI-9113, this information is very important for the programmers who want to control the hardware by low-level programming.
- **Chapter 5**, "Operation Theorem" describes how to operate the PCI-9113. The A/D and timer/counter functions are introduced. Also, some programming concepts are specified.
- **Chapter 6**, "C/C++ Software Library" describes high-level programming interface in C/C++ language. It helps programmer to control PCI-9113 in high level language style.
- **Chapter 7**, "Calibration & Utility," describes how to calibrate the
- **Chapter 8**, "Software Utility," describes how to use the utility programs included in the software CD.



# Introduction

The PCI-9113 is an advanced data acquisition card based on the 32-bit PCI Bus architecture. High performance designs and the state-of-the-art technology make this card ideal for data logging and signal analysis applications in medical, process control, and etc.

The outstanding features of PCI-9113 is that isolated programmable amplifier is used between analog input lines and A/D converter. It can protect all of the A/D converters, your PC, and peripherals from damage due to the high voltages on the analog inputs. The block diagram of PCI-9113 is shown below.



---

## 1.1 Software Supporting

There are several software options to help you to implement your applications quickly and easily.

### ***Custom Program***

For the customers writing their own programs, the PCI-9113 is supported by a comprehensive set of drivers and programming tools. These software drivers support multiple platforms.

- MS-DOS Borland C++ programming library.
- Dynamic linking library (DLL) for Win-95.
- PCIS-DASK/NT : Advanced data acquisition software kit for Windows NT.

The MS-DOS Borland C++ library and DLL for Win-95 are included in ADLink's "Manual & Software Utility" CD in this package.

---

## 1.2 Features

The PCI-9113 PCI Bus Advanced Data Acquisition Card provides the following advanced features:

- 32-bit PCI-Bus, Plug and Play
- 32-CH 12-bit single-ended analog inputs
- Isolation 1500Vrms continuous, 2500Vrms for one minute
- Programmable gain of 1, 10, 100
- Sampling rate up to 60KHz
- Trigger mode: software trigger, timer pacer, external trigger
- On-board A/D 1K WORDS FIFO memory
- Auto-scanning channel selection

- DB-37 connector, pin assignment is compatible with ACL-8113
- Input impedance: 10M  $\Omega$
- Voltage protection: 70 voltage (peak-to-peak)

---

## 1.3 Applications

- Industrial process control
- Transducer, thermocouple, RTD
- Power monitor
- Medical instrument
- Biomedical measurement
- Ground loop elimination

---

## 1.4 Specifications

- ◆ Analog Input (A/D)
  - **Converter** : B.B. ADS774, successive approximation type
  - **Resolution** : 12-bit
  - **Input channels** : 32 single-ended
  - **Isolated programmable gain amplifier**  
**Isolation voltage rated continuous:**  
 1,500Vrms, 2,500Vrms for one minute  
**Isolation-Mode Rejection:** 115dB  
**Barrier Impedance:** 10M $\Omega$  || 10 pF  
**Leakage Current:** 0.8  $\mu$  Ams
  - **Input Range** : (Software controlled, and jumper selection)  
 Bipolar :  $\pm 10V$ ,  $\pm 1V$ ,  $\pm 0.1V$  or  $\pm 5V$ ,  $\pm 0.5V$ ,  $\pm 0.05V$

Unipolar : 0~10V, 0~1V, 0~0.1V



- **Throughput:**  
 Gain = 1            60K  
 Gain = 10          60K  
 Gain = 100        10K
- **Overvoltage Protection:** Continuous  $\pm 35\text{V}$  maximum
- **Accuracy:** 0.015% of reading  $\pm 1$  bit
- **Input Impedance:** 10 M $\Omega$
- **Trigger Mode:** Software and Pacer
- **Data Transfer:** Program control, Interrupt
- **FIFO Buffer Size:** 1024 samples

◆ General Specifications

- **Connector :** 37-pin D-type connector
- **Operating Temperature :** 0° C ~ 55° C
- **Storage Temperature :** -20° C ~ 80° C
- **Humidity :** 5 ~ 95%, non-condensing
- **Power Consumption :** +5 V @ 960mA (max.)
- **Dimension :** 6.77" (172mm) (L) x 4.13" (105mm) (W)



# 2

## Installation

This chapter describes how to install the PCI-9113. At first, the contents in the package and unpacking information that you should be careful are described.

---

### 2.1 What You Have

In addition to this *User's Manual*, the package includes the following items:

- PCI-9113 Enhanced Multi-function Data Acquisition Card
- Manual & Software Utility CD

If any of these items is missing or damaged, contact the dealer from whom you purchased the product. Save the shipping materials and carton in case you want to ship or store the product in the future.

---

## 2.2 Unpacking

Your PCI-9113 card contains sensitive electronic components that can be easily damaged by static electricity.

The card should be done on a grounded anti-static mat. The operator should be wearing an anti-static wristband, grounded at the same point as the anti-static mat.

Inspect the card module carton for obvious damage. Shipping and handling may cause damage to your module. Be sure there are no shipping and handling damages on the module before processing.

After opening the card module carton, extract the system module and place it only on a grounded anti-static surface component side up.

Again inspect the module for damage. Press down on all the socketed IC's to make sure that they are properly seated. Do this only with the module place on a firm flat surface.

---

**Note: DO NOT APPLY POWER TO THE CARD IF IT HAS BEEN DAMAGED.**

---

***You are now ready to install your PCI-9113.***

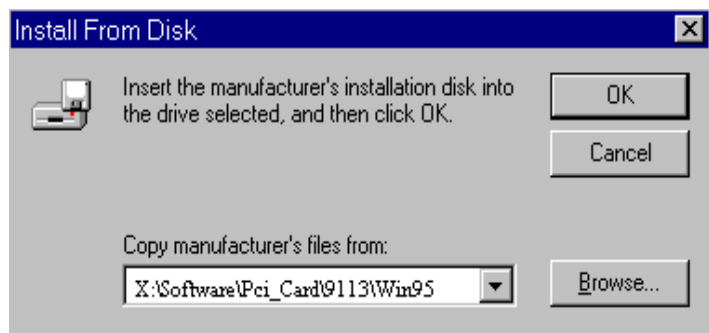
---

## 2.3 Device Installation for Windows 95

While you first plug PCI-9113 card and enter Windows 95, the system will detect this device automatically and show the following dialog box that prompts you to select the device information source.



Choose the default option “*Driver from disk provided by hardware manufacturer*” and then a dialog box is shown to prompt you give the path of installation disk.

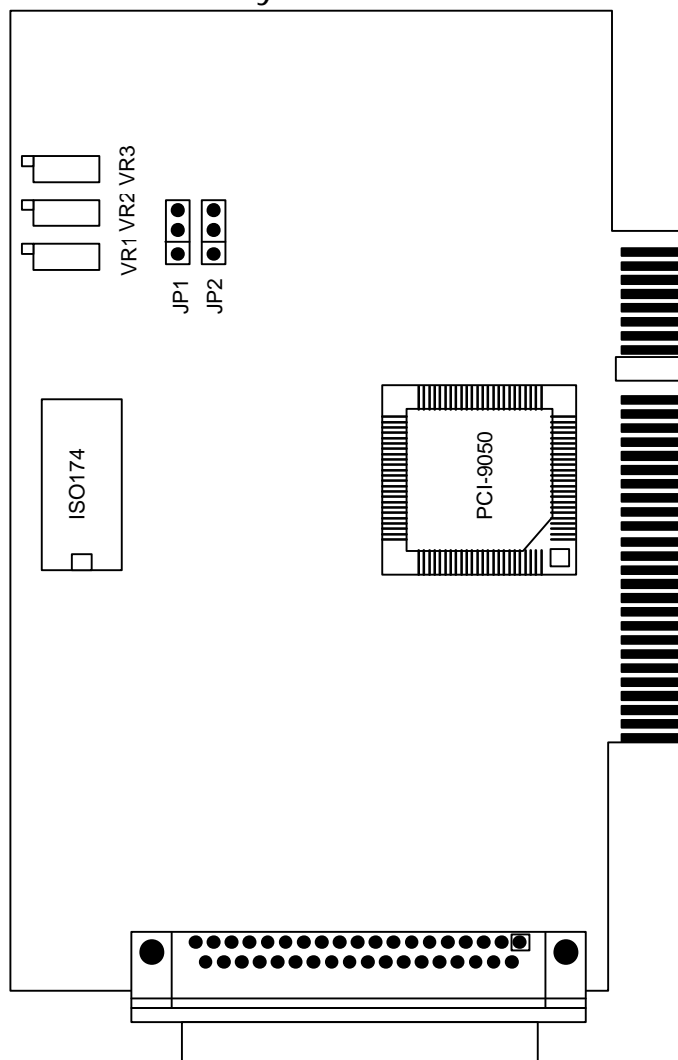


Place ADLink’s “Manual & Software Utility” CD into the appropriate CD driver. Type “X:\Software\Pci\_Card\9113\Win95” (this directory includes PCI-9113 device information file “9113.inf”) in the input field (**X indicates the CD ROM driver**)

and then click OK. The system will start the installation of PCI-9113.

---

## 2.4 PCI-9113's Layout



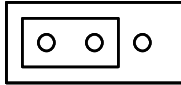
**Figure 2.1** PCB Layout of the PCI-9113

---

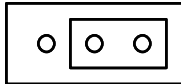
## 2.5 Jumper Descriptions

### 2.5.1 Polarity Selection Jumper

JP1 is the polarity selection jumper. The following diagram shows the possible configurations.



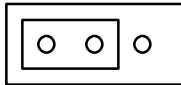
Bi-polar input



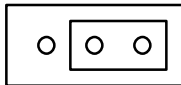
Uni-polar input

### 2.5.2 Full Range Jumper

JP2 set the full range of the analog input channels. The following diagram shows the possible configurations.



20V Full Range



10V Full Range

### 2.5.3 Possible AD Input Range Configurations

The JP1 and JP2 are used to setup the analog input signal range. There are three possible combinations, 0~10V, -5V~+5V and -10V ~ +10V. See the following table for reference.

		JP2	
		2-3	1-2
JP1	1-2	+/-5V	+/-10V
	2-3	0~10V	X



---

## 2.6 PCI Configuration

### **1. Plug and Play :**

As a plug and play component, the board requests an interrupt via a system call. The system BIOS assigns an interrupt level based on the board information and system parameters. These system parameters are determined by the installed drivers and the hardware load seen by the system.

### **2. Configuration :**

The board configuration is done on a board-by-board basis for all PCI cards on your system. Because the configuration is controlled by the system BIOS and software, there is no jumpers for setting system parameters like base-address, and interrupt level.

The configuration is subject to change with every boot of the system when new boards are added or boards are removed.

### **3. Trouble shooting :**

If your system will not boot or if you experience erratic operation with your PCI board in place, it's likely caused by an interrupt conflict (perhaps because you incorrectly described the ISA setup). In general, the solution, once you determine it is not a simple oversight, is to consult the BIOS documentation that come with your system.



# 3

## Signal Connections

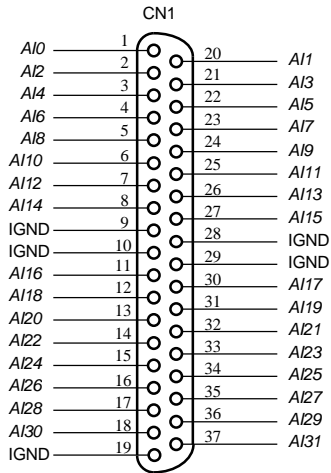
This chapter describes the connector of PCI-9113, and the signal connection between the PCI-9113 and external devices, such as daughter boards or other devices.

---

### 3.1 Connectors Pin Assignment

The PCI-9113 comes equipped one 37-pin D-type connector - CN1. The pin assignment of CN1 is illustrated in the Figure 3.1.

## • CN 1 : Analog Input Signals



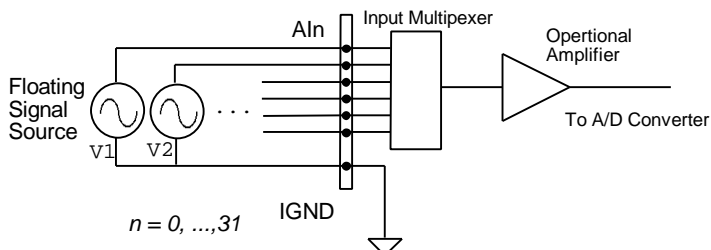
**Figure 3.1 Pin Assignment of CN1**

### Legend :

$AI_n$  : Analog Input Channel  $n$  (single-ended)  
 IGND : Isolated Signal Ground

## 3.2 Analog Input Signal Connection

The PCI-9113 provides 32 single-ended analog input signals which can be converted to digital value by the A/D converter. To avoid ground loops and get more accurate measurement of A/D conversion, it is quite important to understand the signal source type. The single-ended mode has only one input relative to ground and is suitable for connecting with the *floating signal source*. The floating source means it does not have any connection to real ground. Figure 3.2 shows the single-ended connection. Note that when more than two floating sources are connected, the sources must be with common ground.



**Figure 3.2 Floating sources and single-ended connection**

---

### 3.3 Daughter Board Connection

The PCI-9113 can be connected with several different daughter boards, ACLD-9881, ACLD-9137 and ACLD-9188. The functionality and connections are specified as follows.

#### 3.3.1 Connect with ACLD-9881

The ACLD-9881 has a 37-pin D-sub connector, which can connect with PCI-9113 through 37-pin assemble cable. The ACLD-9881 provides low pass filter for the 32 channels, it is very flexible for wiring.

#### 3.3.2 Connect with ACLD-9137

The ACLD-9137 is a direct connector for the card which is equipped with 37-pin D-sub connector. This board provides a simple way for connection. It is very suitable for the simple applications that do not need complex signal condition before the A/D conversion is performed.

#### 3.3.3 Connect with ACLD-9188

ACLD-9188 is general purpose terminal boards for all the card which comes equipped with 37-pin D-sub connector.



## Registers Structure & Format

The detailed descriptions of the register format of the PCI-9113 are specified in this chapter. This information is quite useful for the programmers who wish to handle the card by low-level programming. In addition, users can understand how to use software driver to manipulate this card after understanding the registers' structure of the PCI-9113.

---

### 4.1 I/O Port Address

The PCI-9113 functions as a 32-bit PCI target device to any master on the PCI bus. There are three types of registers on the PCI-9113: PCI Configuration Registers (PCR), Local Configuration Registers (LCR) and PCI-9113 registers.

The PCR which conforms the PCI-bus specifications are initialized and controlled by the system plug & play PCI BIOS. Users can study the PCI BIOS specifications to understand the operation of the PCR. The PCR can only be read through by PCI BIOS function call.

The LCR are specified by the PCI bus controller PLX-9050. It is not necessary for users to understand the details of the LCR if

you use the software library. The base address of the LCR is assigned by the PCI p&p BIOS. The assigned address is located at offset 14h of PCR.

The PCI-9113 registers are shown in the Table 4.1. The base address of the PCI-9113 registers is also assigned by the PCI p&p BIOS. The assigned base address is located at offset 18h of PCR. Note that most of the PCI-9113 registers are 16 bits. The users can access these registers by 16 bits I/O instructions.

There is one 32 bits register on PCI-9113. The 32 bits register occupied another LCR address space, that is, base address #2. The base address is allocated by PCI BIOS and is stored at offset 1Ch of PCR.

Users can read the PCR to get the LCR base address and the two PCI-9113 base addresses by using the PCI BIOS function call.

<b>I/O Base Address #1</b>	<b>Write</b>	<b>Read</b>
Base + 00h	AD MUX channel no.	AD FIFO value
Base + 02h	AD range control	AD status read back
Base + 04h	AD trigger mode	AD trigger mode
Base + 06h	Interrupt control	Interrupt setting Read Back
Base + 08h	Software AD trigger	--
Base + 0Ah	Clear H/W IRQ	--
<b>I/O Base Address #2</b>	<b>Write</b>	<b>Read</b>
Base 2 + 00h	--	AD data and channel number

**Table 4.1 I/O Address**

---

## 4.2 A/D Data Registers

The PCI-9113 A/D data is stored in the FIFO after conversion. The data can be transferred to host memory by software only. The register is 12 bits and can be read by 16 bits I/O command.



**Address : BASE + 0**

**Attribute :** read only

**Data Format :**

Bit	7	6	5	4	3	2	1	0
BASE+0	AD7	AD6	AD5	AD4	AD3	AD2	AD1	AD0
BASE+1	x	x	x	x	AD11	AD10	AD9	AD8

AD11 .. AD0 : Analog to digital data. AD11 is the Most Significant Bit (MSB) of PCI-9113. AD0 is the Least Significant Bit(LSB).

---

## 4.3 A/D Channel Control Register

The PCI-9113 provides 32 single-ended analog input channels. The channel control register is used to set the A/D channels to be converted. The 5 LSBs of this register control the channel number. Under non-auto scanning mode, the register sets the channel number for conversion. Under auto-scanning mode, the register set the ending channel number.

**Address : BASE + 0**

**Attribute :** write only

**Data Format :**

Bit	7	6	5	4	3	2	1	0
BASE+0	x	x	x	CN4	CN3	CN2	CN1	CN0
BASE+1	x	x	x	x	x	x	x	x

CNn : multiplexer channel number.

CN4 is MSB, and CN0 is LSB.

---

## 4.4 A/D Input Signal Range Control Register

The A/D range register is used to adjust the analog input ranges. This register directly controls the PGA (programmable gain amplifier). When a different gain value is set, the analog input range will be changed to its corresponding value.

**Address : BASE + 2**

**Attribute :** write only

**Data Format :**

Bit	7	6	5	4	3	2	1	0
BASE+2	X	X	X	X	X	X	G1	G0
BASE+3	X	X	X	X	X	X	X	X

GC0~GC1: A/D Range control setting

The relationship between gain setting and its corresponding A/D range is listed in the table below.

G2	G1	GAIN	Analog Input Range		
			Bi-Polar		Uni-Polar
0	0	1	±10V	±5V	10 V
0	1	10	±1V	±500mV	1V
1	0	100	±100mV	±50mV	100mV

---

## 4.5 A/D Status Readback Register

The A/D FIFO status can be read back from this register.

**Address :** BASE + 2

**Attribute :** read only

**Data Format :**

Bit	7	6	5	4	3	2	1	0
BASE+2	X	X	X	X	AD_BUSY	FF_FF	FF_HF	FF_EF
BASE+3	X	X	X	X	X	X	X	X

FF\_EF : '0' means FIFO is empty

FF\_HF : '0' means FIFO is half-full

FF\_FF : '0' means FIFO is full, A/D data may have been loss

AD\_BUSY : '0' means AD is busy, the A/D data has not been latched in FIFO yet. If AD\_BUSY changes from '0' to '1', A/D data is written into FIFO.

---

## 4.6 A/D Trigger Mode Control and Readback Register

This register is used to control or read back the A/D trigger control setting and the A/D range setting.

**Address : BASE + 4**

**Attribute :** write and read

**Data Format :**

Bit	7	6	5	4	3	2	1	0
BASE+4	X	X	X	X	G1	G0	TSSEL	ASCAN
BASE+5	X	X	X	X	X	X	X	X

GC0~GC1: A/D range setting, read back (only)

TSSEL : Timer Pacer / Software Trigger

1: Timer Pacer Trigger

0: Software Trigger

ASCAN: Auto Scan Control

1: Auto Scan ON

0: Auto Scan OFF

## 4.7 Software Trigger Register

To generate a trigger pulse to the PCI-9113 for A/D conversion, you just write any data to this register, then the A/D converter will be triggered.

**Address : BASE + 8**

**Attribute :** write only

**Data Format :**

Bit	7	6	5	4	3	2	1	0
BASE+8	X	X	X	X	X	X	X	X

## 4.8 Interrupt Control and Readback Register

The PCI-9113 has a dual interrupt system, thus two interrupt sources can be generated and be checked by the software. This register is used to select the interrupt sources.

**Address : BASE + 6**

**Attribute :** write and read

**Data Format :**

Bit	7	6	5	4	3	2	1	0
BASE+12	X	X	X	X	X	FFEN	ISC1	ISC0

ISC0 : IRQ0 signal select

0 : IRQ on the ending of the AD conversion (EOC)

1 : IRQ when FIFO is half full

ISC1 : IRQ1 signal select (Timer Interrupt only)

FFEN : FIFO enable pin

0: FIFO Enable (Power On Default value)

1: FIFO Disable

(To reset FIFO, set FFEN sequence as 0 -> 1 -> 0)

---

## 4.9 Hardware Interrupt Clear Register

Because the PCI interrupt signal is level trigger, the interrupt clear register must be written to clear the flag after processing the interrupt request event; otherwise, that another interrupt request is inserted will cause the software to hang on processing the interrupt event.

**Address :** BASE + 0Ah

**Attribute :** write only

**Data Format :**

Bit	7	6	5	4	3	2	1	0
BASE+0Ah	X	X	X	X	X	X	X	X

---

## 4.10 A/D Data and Channel Number Registers

The PCI-9113 A/D data and channel number is stored in the FIFO. Reading this register by a 32-bit I/O instruction can read back the data and channel number simultaneously.

**Address :** BASE 2 + 0

**Attribute :** read only

**Data Format :**

Bit	7	6	5	4	3	2	1	0
BASE2+0	AD7	AD6	AD5	AD4	AD3	AD2	AD1	AD0
BASE2+1	x	X	x	x	AD11	AD10	AD9	AD8
BASE2+2	x	X	x	CN4	CN3	CN2	CN1	CN0
BASE2+3	x	X	x	x	x	x	x	x

AD11~AD0 : Analog to digital data. AD11 is the Most Significant Bit (MSB) of PCI-9113. AD0 is the Least Significant Bit (LSB).

CN4~CN0 : Channel number

---

## 4.11 High Level Programming

To operate the PCI-9113, you can bypass the detailed register structures and use the high-level application programming interface (API) to control your PCI-9113 card directly. The software libraries, DOS library for Borland C++, and DLL for Windows 95 are included in the ADLink's "Manual & Software Utility" CD. Please refer to chapter 6 for more detailed information.

---

## 4.12 Low-Level Programming

To operate the PCI-9113, users do not need to understand how to write a hardware dependent low-level program. The control of the PCI controller is complex and not described in the manual. It is not recommended the user to program applications based on low-level programming. If the user does need to program in low-level programming, you can contact the dealer from whom you purchased the PCI-9113 for further PCI controller programming information.



# Operation Theorem

The operation theorem of the functions on PCI-9113 card is described in this chapter. The operation theorem can help you to understand how to manipulate or to program the PCI-9113.

---

## 5.1 A/D Conversion

Before programming the PCI-9113 to perform the A/D conversion, you should understand the following issues:

- A/D conversion procedure
- A/D signal source control
- A/D trigger source control
- A/D data transfer mode
- Interrupt System (refer to section 5.2)
- A/D data format

---

**Note :** Because some of the A/D data transfer modes will use the system interrupt resource, the users have to understand the interrupt system (section 5.2) in the same time.

---

### **5.1.1 A/D Conversion Procedure**

For using the A/D converter, users must know about the property of the signal to be measured at first. The users can decide which channels to be used and connect the signals to the PCI-9113. Refer to the chapter 3 'Signal Connection'. In addition, users should define and control the A/D signal sources, including the A/D channel, A/D gain, and A/D signal types. Please refer to section 5.1.2. for A/D signal source control.

After deciding the A/D signal source, the user must decide how to trigger the A/D conversion and define/control the trigger source. The A/D converter will start to convert the signal to a digital value when a trigger signal is rising. Refer to the section 5.1.3 for the two trigger sources.

The A/D data should be transferred into PC's memory for further using or processing. The data can be read by I/O instruction which is handled directly by software or transferred to memory via interrupt. Please refer to section 5.1.4 to obtain ideas about the multi-configurations for A/D data transfer.

To process A/D data, programmer should know about the A/D data format. Refer to section 5.1.5 for details.

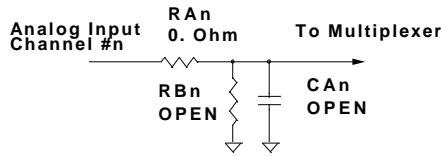
### **5.1.2 A/D Signal Source Control**

To control the A/D signal source, the signal type, signal channel and signal range should be considered.

#### ***Signal Type & Signal Conditioning***

The A/D signal sources of PCI-9113 could be single ended (SE) only. There are 32 SE A/D channels on board. The R/C filters (attenuators) are on board for every channel. The RC circuits for each channel is shown in the following diagram, where 'n' is the channel number. User can install the R, C for special purpose such as attenuating the voltage to increase the input voltage range.





The RC network can also be used as current sensor, which transfers the current into voltage. To get the ground reference level, user can cut-off RA and let RB = 0 Ohm, thus grounding the input signal. The users can use the ground input to calibrate the offset voltage by software.

### ***Signal Channel Control***

There are two ways to control the channel number. The first one is the software programming and the second one is the auto channel scanning which is controlled by the ASCAN bit in AD trigger mode control register. As ASCAN is cleared (0), the value of AD channel Control register defines the channel to be selected.

As ASCAN is set 1, the value in AD channel control register defines the ending channel number of auto-scanning operation. Under auto scan mode, the channel is scanning from channel 0 to the ending channel. Whenever a trigger signal is rising, the channel number to be selected will increase automatically. For example, if the ending channel number is 3, the auto channel scanning sequence is 0,1,2,3,0,1,2,3, ..., until the ASCAN bit is cleared.

### ***Signal Range***

The proper signal range is important for data acquisition. The input signal may be saturated if the A/D gain is too large. Sometimes, the resolution may be not enough if the signal is small. The maximum A/D signal range of PCI-9113 is  $\pm 10$  volts when the A/D gain value is 1. The A/D gain control register controls the maximum signal input range. The signal gain is

programmable with 4 levels (1,10,100,1000\*). The signal range of the 32 channels will be identical all the time even if the channel number is scanning. The available signal polarity on PCI-9113 are bi-polar and uni-polar configuration.

---

**\*Note:** Gain value of 1000 is programmable, however, the accuracy is not guaranteed.

---

### 5.1.3 A/D Trigger Source Control

The A/D conversion is started by a trigger source, and then the A/D converter will start to convert the signal to a digital value. In PCI-9113, two internal sources can be selected: the software trigger or the timer pacer trigger. The A/D operation mode is controlled by A/D trigger mode register. Total two trigger sources are provided in the PCI-9113. The different trigger conditions are specified as follows:

#### ***Software trigger (TSSEL=0)***

The trigger source is software controllable in this mode. That is, the A/D conversion is starting when any value is written into the software trigger register. This trigger mode is suitable for low speed A/D conversion. Under this mode, the timing of the A/D conversion is fully controlled by software. However, it is difficult to control the fixed A/D conversion rate unless another timer interrupt service routine is used to generate a fixed rate trigger. Refer to interrupt control section (section 5.2) for fixed rate timer interrupt operation.

#### ***Timer Pacer Trigger (TSSEL=1)***

An on-board timer / counter chip 8254 is used to provide a trigger source for A/D conversion at a fixed rate. Two counters of the 8254 chip are cascaded together to generate trigger pulse with precise period. Please refer to section 5.3 for 8254 architecture. This mode is ideal for high speed A/D conversion. It can be combined with the FIFO half-full interrupt or EOC interrupt to transfer data. It is also possible to use software FIFO polling to

transfer data. The A/D trigger, A/D data transfer and Interrupt can be set independently. Most of the complex applications can thus be covered.

It's recommended using this mode if your applications need a fixed and precise A/D sampling rate.

#### **5.1.4 A/D Data Transfer Modes**

The A/D data are buffered in the FIFO memory. The FIFO size on PCI-9113 is 1024 (1K) words. If the sampling rate is 10 KHz, the FIFO can buffer 102.4 ms analog signal. After the FIFO is full, the lasting coming data will be lost. The software must read out the FIFO data before it becomes full.

The data must be transferred to host memory after the data is ready and before the FIFO is full. On the PCI-9113, many data transfer modes can be used. The different transfer modes are specified as follows:

##### ***Software Data Polling***

The software data polling is the easiest way to transfer A/D data. This mode can be used with software A/D trigger mode. After the A/D conversion is triggered by software, the software should poll the *FF\_EF* bit of the A/D status register until it becomes low level.

If the FIFO is empty before the A/D start, the *FF\_EF* bit will be low. After the A/D conversion is completed, the A/D data is written to FIFO immediately, thus the *FF\_EF* becomes high. You can consider the *FF\_EF* bit as a flag to indicate the converted data ready status. That is, *FF\_EF* is high means the data is ready. Note that, while A/D is converted, the *ADBUSY* bit is low. After A/D conversion, the *ADBUSY* becomes high to indicate not busy. Please do NOT use this bit to poll the AD data.

It is possible to read A/D converted data without polling. The A/D conversion time will not exceed 8.5 $\mu$ s on PCI-9113 card. Hence,

after software trigger, the software can wait for at least 8.5 $\mu$ s and then read the A/D register without polling.

The data polling transfer is very suitable for the application that needs to process AD data in real time. Especially, when combining with the timer interrupt generation, the timer interrupt service routine can use the data polling method to get multi-channel A/D data in real time and with the fixed data sampling rate.

### ***FIFO Half-Full Polling***

The FIFO half-full polling mode is the most powerful AD data transfer mode. The 1 K words FIFO can be stored up to 10.24 ms analog data under 100 KHz sampling rate ( $10.024\text{ms} = 1024 / 100 \text{ KHz}$ ). Theoretically, the software can poll the FIFO every 10 ms without taking care how to trigger A/D or transfer A/D data.

It's recommend that users check your system to find out the user software's priority in the special application. If the application software is at the highest priority, polling the FIFO every 10 ms is suitable. However, the user's program must check the FIFO is full or empty every time reading data.

To avoid this problem, the half-full polling method is used. If the A/D trigger rate is 100KHz, the FIFO will be half-full (512 words) in 5.12 ms. If the user's software checks the FIFO half full signal every 5 ms and the FIFO is not half-full, the software does not read data. When the FIFO is full, the AD FIFO is overrun. That means the sampling rate is higher than users' expect or the polling rate is too slow. It is also possible due to your system occupy the CPU resource thus reducing the polling rate. When the FIFO is half-full and not full, the software can read one "block" (512 words) A/D data without checking the FIFO status. This method is very convenient to read A/D in size of a "block" and it is benefit to software programming.

Usually, the timer trigger is used under this mode, therefore the sampling rate is fixed. The method also utilizes the minimum CPU resources because it is not necessary to be the highest priority. The other benefit is this method will not use hardware interrupt resource. Therefore, the interrupt is reserved for system clock or emergency external interrupt request. The FIFO half-full polling method is the most powerful A/D data transfer mode.

### ***EOC Interrupt Transfer***

The PCI-9113 provides traditional hardware end-of-conversion (EOC) interrupt capability. Under this mode, an interrupt signal is generated when the A/D conversion is ended and the data is ready to be read in the FIFO. It is useful to combine the EOC interrupt transfer with the timer pacer trigger mode. After A/D conversion is completed, the hardware interrupt will be inserted and its corresponding ISR (Interrupt Service Routine) will be invoked and executed. The ISR program can read the converted data. This method is most suitable for data processing applications under real-time and fixed sampling rate

### ***FIFO Half-Full Interrupt Transfer***

Sometimes, the applications do not need real-time processing, but the foreground program is too busy to poll the FIFO data. The FIFO half-full interrupt transfer mode is useful for the situation mentioned above. In addition, as the external A/D trigger source is used, the sampling rate may be not easy to predict, and then the method could be applied. Because the CPU is only interrupted when the FIFO is half-full, thus reserved the CPU load.

Under this mode, an interrupt signal is generated when FIFO becomes half-full. It means there are 512 words data in the FIFO already. The ISR can read a block of data every interrupt occurring. This method is very convenient to read A/D in size of a "block" (512 words) and it is benefit for software programming.

### 5.1.5 A/D Data Format

The range of A/D data read from the FIFO port is from 0 to 4095. As the A/D gain is 1, the A/D signal range is roughly -10V ~ +10V or -5V~+5V (bi-polar) and 0V~+10V (uni-polar). The relationship between the voltage and the value is shown in the following table:

A/D Data (Hex)	Decimal Value	Voltage (Volts)	
		±10V (Bipoar)	0~10V(Unipoar)
FFF	4095	+9.9951	+9.9951
C00	3072	+5.0000	+7.5000
801	2049	+0.0049	+5.0049
800	2048	0.0000	+5.0000
7FF	2047	-0.0049	+4.9951
400	1024	-5.0000	+2.5000
000	0	-0.0000	0.0000

The formula between the A/D data and the analog value is

$$\begin{aligned} \text{Voltage} &= ( \text{AD\_Data} * 20 ) / ( 4096 * \text{gain} ) - ( 10/\text{gain} ) \quad \text{--} \\ &\quad \text{Bipolar} \\ \text{Voltage} &= ( \text{AD\_Data} * 10 ) / ( 4096 * \text{gain} ) \quad \text{--} \\ &\quad \text{Unipolar} \end{aligned}$$

where the *gain* is 1,10,100.

---

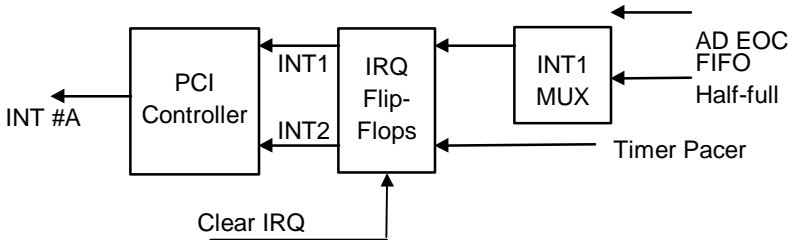
## 5.2 Interrupt Control

### 5.2.1 System Architecture

The PCI-9113's interrupt system is a powerful and flexible system that is suitable for A/D data acquisition and many applications. The system is a **Dual Interrupt System**. The dual interrupt means the hardware can generate two interrupt request signals in the same time and the software can service these two request signals by ISR. Note that the dual interrupt does not mean the

card occupies two IRQ levels.

The two interrupt request signals (INT1 and INT2) come from digital signals or the timer / counter output. An interrupt source multiplexer (MUX) is used to select the IRQ sources. Fig 5.2.1 shows the interrupt system.



**Fig 5.2.1 Dual Interrupt System of PCI-9113**

### 5.2.2 IRQ Level Setting

There is only one IRQ level used by this card, although it is a dual interrupt system. This card uses INT #A interrupt request signal to PCI bus. The motherboard circuits will transfer INT #A to one of the AT bus IRQ levels. The IRQ level is set by the PCI plug and play BIOS and saved in the PCI controller. It is not necessary for users to set the IRQ level.

### 5.2.3 Dual Interrupt System

The PCI controller of PCI-9113 can receive two hardware IRQ sources. However, a PCI controller can generate only one IRQ to PCI bus, the two IRQ sources should be distinguished by ISR of the application software if the two IRQ are all used.

The application software can use the “\_9113\_Get\_Irq\_Status” function to distinguish which interrupt is inserted. After servicing an IRQ signal, users should check if another IRQ is also asserted and then clear current IRQ to allow the next IRQ occurring.

The two IRQs are named as INT1 and INT2. INT1 comes from AD EOC or the FIFO half-full flag. INT2 comes from timer's pacer output only. The sources of INT1 and INT2 are selective by the Interrupt Control (ISC) Register.

Because of dual interrupt system, for example, you can use FIFO half-full and external interrupt at the same time if your software ISR can distinguish these two events.

#### **5.2.4 Interrupt Source Control**

There are two bits to control the IRQ sources of INT1 and INT2. Refer to section 4.9 for the details of the two bits. In addition, the PCI controller itself can also control the use of the interrupt. For manipulating the interrupt system more easily, ADLink recommends you to use the function `_9113_INT_Source_Control` to control the IRQ source so that you can disable one or two of the IRQ sources.

Note that even you disable all the two IRQ sources without changing the initial condition of the PCI controller, the PCI BIOS still assigns an IRQ level to the PCI card and it will occupy the PC resource. It is not suggested to re-design the initial condition of the PCI card by users' own application software. If users want to disable the IRQ level, please use the ADLink's software utility to change the power on interrupt setting.

---

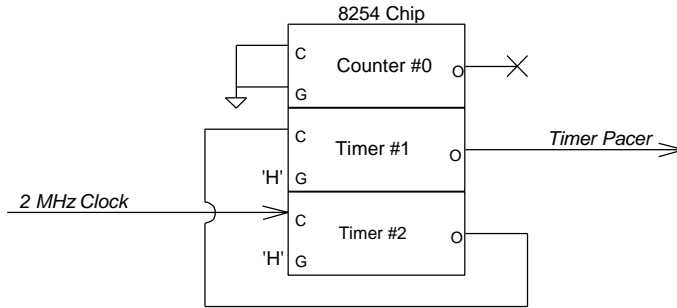
### **5.3 Timer/Counter Operation**

The PCI-9113 has an interval timer/counter 8254 on board. Refer to section 3.5 for the signal connection and the configuration of the counters.



### 5.3.1 Introduction

One 8254 programmable timer/counter chip is installed in PCI-9113. There are three counters in one 8254 chip and 6 possible operation modes for each counter. The block diagram of the timer/counter system is shown in following diagram.



**Figure 5.3.1 Timer/Counter System of PCI-9113.**

### 5.3.2 Pacer Trigger Source

The timer #1 and timer #2 are cascaded together to generate the timer pacer trigger of A/D conversion. The frequency of the pacer trigger is software controllable. The maximum pacer signal rate is  $2\text{MHz}/4=500\text{K}$  which excess the maximum A/D conversion rate of the PCI-9113 (80KHz). The minimum signal rate is  $2\text{MHz}/65535/65535$ , which is a very slow frequency that user may never use it. The output of the programmable timer can be used as the pacer interrupt source or the timer pacer trigger source of A/D conversion. In software library, the timer #1 and #2 are always set as mode 2 (rate generator) or mode 3.



# 6

## C/C++ Software Library

There are 36 functions provided by the C Language library. This library includes all the functions of PCI-9113. The major capability of these function calls is A/D conversion. In addition, there are some sample programs to help you to use this library.

---

### 6.1 Installation

#### 6.1.1 Installation

##### ◆ MS-DOS Software Installation

**step 1.** Place ADLink's "Manual & Software Utility" CD into the appropriate CD drive.

**step 2.** Type the command (X indicates the CD ROM driver):

X:\> **CD Software\Pci\_Card\9113\DOS**

X:\ Software\Pci\_Card\9113\DOS> **SETUP**

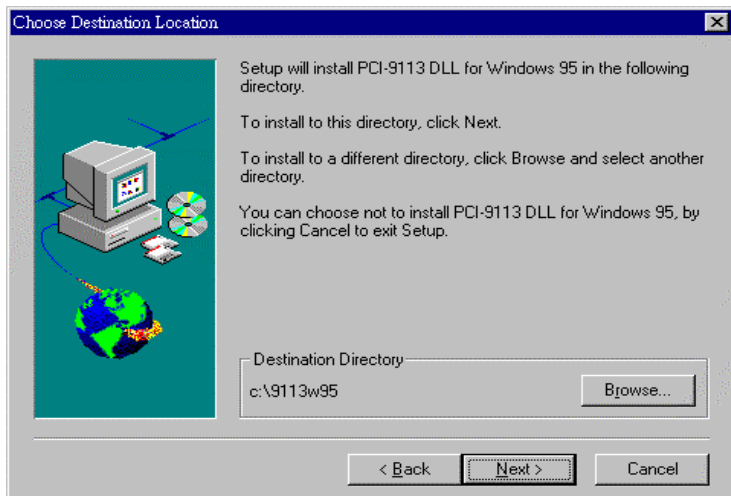
**step 3.** An installation complete message will be shown on the screen.

After installation, all the files of *PCI-9113 Library & Utility for DOS* are stored in C:\ADLink\9113\DOS directory.

◆ **Windows 95 Software Installation**

- step 1.** Place ADLink's "Manual & Software Utility" CD into the appropriate CD driver.
- step 2.** If Windows 95 is loaded, choose Run from the Start menu.
- step 3.** Type **X:\Software\Pci\_Card\9113\Win95\Setup.exe** in the Run dialog box. (X indicates the CD ROM driver).

After a welcome dialog box, Setup prompts the following dialog box for you to specify the destination directory. The default path is C:\9113w95. If you want to install *PCI-9113 DLL for Windows 95* in another directory, please click Browse button to change the destination directory. Then you can click Next to begin installing *PCI-9113 DLL for Windows 95*.



Then you can click Next to begin installing *PCI-9113 DLL for Windows 95*.

After you complete the installation of PCI-9113 Software, PCI-9113's DLL (9113.DLL) is copied to Windows System directory (default is C:\WINDOWS\SYSTEM) and the driver files (W95\_9113.VXD and PCIW95.VXD) are also copied to the appropriate directory.

---

## 6.2 C/C++ Programming Library

We defined some data types in `acl_pci.h`. These data types are used by PCI-9113 library. We suggest you to use these data types in your application programs. The following table shows the data type names and their range.

### 6.2.1 Data Types

Type Name	Description	Range
U8	8-bit ASCII character	0 to 255
I16	16-bit signed integer	-32768 to 32767
U16	16-bit unsigned integer	0 to 65535
I32	32-bit signed integer	-2147483648 to 2147483647
U32	32-bit single-precision floating-point	0 to 4294967295
F32	32-bit single-precision floating-point	-3.402823E38 to 3.402823E38
F64	64-bit double-precision floating-point	-1.797683134862315E308 to 1.797683134862315E309
Boolean	Boolean logic value	TRUE, FALSE

The functions of PCI-9113's software drivers use full-names to represent the functions' real meaning. The naming convention rules are :

In DOS Environment :

`_{hardware_model}_{action_name}`. e.g. `_9113_Initial()`.

In order to recognize the difference between DOS library and Windows 95 library, A capital "W" is put on the head of each

function name of the Windows 95 DLL driver. e.g.  
**W\_9113\_Initial()**.

There are 36 functions provided by PCI-9113 software drivers.  
The detail descriptions of each function are specified in the  
following sections.

### **6.2.2    \_9113\_Initial**

#### **@ Description**

This function is used to initialize PCI-9113 card. Every PCI-9113 card has to be initialized by this function before calling other functions.

#### **@ Syntax**

##### **C/C++ (DOS)**

U16 \_9113\_Initial (U16 \*existCards, PCI\_INFO \*info)

##### **C/C++ (Windows 95)**

U16 W\_9113\_Initial (U16 \*existCards, PCI\_INFO \*info)

##### **Visual Basic (Windows 95)**

W\_9113\_Initial (existCards As Integer, info As PCI\_INFO)  
As Integer

#### **@ Argument**

**existCards:** numbers of existing PCI-9113 cards

**info:** relative information of the PCI-9113 cards

#### **@ Return Code**

ERR\_NoError

ERR\_BoardNoInit

ERR\_PCIBiosNotExist

### **6.2.3    \_9113\_Software\_Reset**

#### **@ Description**

This function is used to reset the I/O port configuration. Note  
that this function can not re-start the PCI bus and all the

hardware setting won't be changed neither.

**@ Syntax**

**C/C++ (DOS)**

void \_9113\_Software\_Reset (U16 cardNo)

**C/C++ (Windows 95)**

void W\_9113\_Software\_Reset (U16 cardNo)

**Visual Basic (Windows 95)**

W\_9113\_Software\_Reset (ByVal cardNo As Integer)

**@ Argument**

**cardNo:** The card number of initialized PCI-9113 card

**@ Return Code**

None

## 6.2.4 \_9113\_AD\_Read\_Data

**@ Description**

This function is used to read the A/D conversion data from A/D Data register. The resolution of A/D conversion data is 12 bits.

**@ Syntax**

**C/C++ (DOS)**

U16 \_9113\_AD\_Read\_Data (U16 cardNo, U16 far  
\*ADData)

**C/C++ (Windows 95)**

U16 W\_9113\_AD\_Read\_Data (U16 cardNo, U16 \*ADData)

**Visual Basic (Windows 95)**

W\_9113\_AD\_Read\_Data (ByVal cardNo As Integer,  
ADData As Integer) As Integer

**@ Argument**

**cardNo:** The card number of initialized PCI-9113 card.

**ADData:** A/D converted value. The resolution of AD data is 12-bit. The bit 0 of ADData is the LSB of A/D converted data and the bit 11 of ADData is the MSB

of A/D converted data. Please refer to section 5.1.5 for the relationship between the voltage and the digital value.

**@ Return Code**

ERR\_NoError

## **6.2.5    \_9113\_AD\_Read\_Data\_Repeat**

**@ Description**

This function is used to read the A/D conversion data from the data register n times continuously.

**@ Syntax**

**C/C++ (DOS)**

U16 \_9113\_AD\_Read\_Data\_Repeat (U16 cardNo, I16 far \*ADDData, U16 n)

**C/C++ (Windows 95)**

U16 W\_9113\_AD\_Read\_Data\_Repeat (U16 cardNo, I16 \*ADDData, U16 n)

**Visual Basic (Windows 95)**

W\_9113\_AD\_Read\_Data\_Repeat (ByVal cardNo As Integer, ADDData As Integer, ByVal n As Integer) As Integer

**@ Argument**

**cardNo:** The card number of PCI-9113 card initialized

**ADDData:** A/D converted value. The resolution of A/D data is 12-bit. The bit 0 of ADDData is the LSB of A/D converted data and the bit 11 of ADDData is the MSB of A/D converted data. Please refer to section 5.1.5 for the relationship between the voltage and the value.

**n:** The number of times to read the A/D conversion data.

**@ Return Code**

ERR\_NoError



## 6.2.6 \_9113\_AD\_Read\_Data\_MUX

### @ Description

This function is used to read data from A/D Data and Channel Number Register. The A/D Data and Channel Number Register is a 32-bit register. Please refer to section 4.10 for the description of A/D Data and Channel Number Register.

### @ Syntax

#### C/C++ (DOS)

U16 \_9113\_AD\_Read\_Data\_MUX (U16 cardNo, U32 far \*ADDData )

#### C/C++ (Windows 95)

U16 W\_9113\_AD\_Read\_Data\_MUX (U16 cardNo, U32 \*ADDData )

#### Visual Basic (Windows 95)

W\_9113\_AD\_Read\_Data\_MUX (ByVal cardNo As Integer, ADDData As Long) As Integer

### @ Argument

**cardNo:** The card number of PCI-9113 card initialized.

**ADDData:** A/D converted value. The resolution of A/D conversion data is 12 bits. The unsigned integer data format of ADDData is as follows:

Every 32-bit unsigned integer data:

bit 0...11: A/D converted data

bit 16, 17, ..., 20 : converted channel no.

Please refer to section 5.1.5 for the relationship between the voltage and the value.

### @ Return Code

ERR\_NoError

## 6.2.7 \_9113\_AD\_Read\_Data\_Repeat\_MUX

### @ Description

This function is used to read data from A/D Data and Channel Number Register n times continuously. The A/D Data and

Channel Number Register is a 32-bit register. Please refer to section 4.10 for the description of A/D Data and Channel Number Register.

**@ Syntax**

**C/C++ (DOS)**

U16 \_9113\_AD\_Read\_Data\_Repeat\_MUX (U16 cardNo,  
U32 far \*ADDData, U16 n)

**C/C++ (Windows 95)**

U16 W\_9113\_AD\_Read\_Data\_Repeat\_MUX (U16 cardNo,  
U32 \*ADDData, U16 n)

**C/C++ (Windows 95)**

W\_9113\_AD\_Read\_Data\_Repeat\_MUX (ByVal cardNo As  
Integer, ADDData As Long, ByVal n As Integer) As  
Integer

**@ Argument**

**cardNo:** The card number of PCI-9113 card initialized

**ADDData:** A/D converted value. The resolution of A/D conversion data is 12 bits. The unsigned integer data format of ADDData is as follows:

Every 32-bit unsigned integer data:

bit 0...11: A/D converted data

bit 16, 17, ..., 20 : converted channel no.

Please refer to section 5.1.5 to learn the relationship between the voltage and the value.

**n:** The timer of times to read the AD conversion data.

**@ Return Code**

ERR\_NoError

## **6.2.8 \_9113\_AD\_Set\_Channel**

**@ Description**

This function is used to set A/D channel by means of writing data to the channel control register. There are 32 single-ended A/D channels in PCI-9113. Therefore the channel number could be set between 0 to 31. Under non-auto scan mode, the

ADChannelNo stores the channel number setting. Under auto-scan mode, the ADChannelNo records the channel number of ending channel.

**@ Syntax**

**C/C++ (DOS)**

U16 \_9113\_AD\_Set\_Channel (U16 cardNo, U16  
ADChannelNo)

**C/C++ (Windows 95)**

U16 W\_9113\_AD\_Set\_Channel (U16 cardNo, U16  
ADChannelNo)

**Visual Basic (Windows 95)**

W\_9113\_AD\_Set\_Channel (ByVal cardNo As Integer,  
ByVal ADChannelNo As Integer) As Integer

**@ Argument**

**cardNo:** The card number of PCI-9113 card initialized.

**ADChannelNo:** The selected channel number or the ending channel number to perform A/D conversion.

**@ Return Code**

ERR\_NoError

## 6.2.9 \_9113\_AD\_Set\_Range

**@ Description**

This function is used to set the A/D range by means of writing data to the A/D range control register. The initial value of gain is '1' which is the default setting by PCI-9113 hardware. The relationship between gain and input voltage ranges is specified by the following tables:

Input Range (V)	Gain	Gain Code
±10 V	X 1	AD_B_10_V
±1 V	X 10	AD_B_1_V
±100m V	X 100	AD_B_0_1_V
±10m V	X 1000	AD_B_0_01_V

±5 V	X 1	AD_B_5_V
±500m V	X 10	AD_B_0_5_V
±50m V	X 100	AD_B_0_05_V
±5m V	X 1000	AD_B_0_005_V
0~10 V	X 1	AD_U_10_V
0~1 V	X 10	AD_U_1_V
0~100m V	X 100	AD_U_0_1_V
0~10m V	X 1000	AD_U_0_01_V

#### @ Syntax

##### **C/C++ (DOS)**

U16 \_9113\_AD\_Set\_Range (U16 cardNo, U16 ADRange)

##### **C/C++ (Windows 95)**

U16 W\_9113\_AD\_Set\_Range (U16 cardNo, U16  
ADRange)

##### **Visual Basic (Windows 95)**

W\_9113\_AD\_Set\_Range (ByVal cardNo As Integer, ByVal  
ADRange As Integer) As Integer

#### @ Argument

**cardNo:** The card number of PCI-9113 card initialized.

**ADRange:** The programmable gain of A/D conversion, the possible values are: AD\_B\_10\_V, AD\_B\_1\_V, AD\_B\_0\_1\_V, AD\_B\_0\_01\_V, AD\_B\_5\_V, AD\_B\_0\_5\_V, AD\_B\_0\_05\_V, AD\_B\_0\_005\_V, AD\_U\_10\_V, AD\_U\_1\_V, AD\_U\_0\_1\_V, AD\_U\_0\_01\_V

#### @ Return Code

ERR\_NoError

### 6.2.10 \_9113\_AD\_Get\_Range

#### @ Description

This function is used to get the A/D range from the A/D range control register. The following table specifies the relationship between the gain and input voltage ranges:

Input Range (V)	Gain	Gain Code
±10 V	X 1	AD_B_10_V
±1 V	X 10	AD_B_1_V
±100m V	X 100	AD_B_0_1_V
±10m V	X 1000	AD_B_0_01_V
±5 V	X 1	AD_B_5_V
±500m V	X 10	AD_B_0_5_V
±50m V	X 100	AD_B_0_05_V
±5m V	X 1000	AD_B_0_005_V
0~10 V	X 1	AD_U_10_V
0~1 V	X 10	AD_U_1_V
0~100m V	X 100	AD_U_0_1_V
0~10m V	X 1000	AD_U_0_01_V

#### @ Syntax

##### C/C++ (DOS)

U16 \_9113\_AD\_Get\_Range (U16 cardNo, U16 \*ADRange)

##### C/C++ (Windows 95)

U16 W\_9113\_AD\_Get\_Range (U16 cardNo, U16  
\*ADRange)

##### Visual Basic (Windows 95)

W\_9113\_AD\_Get\_Range (ByVal cardNo As Integer,  
ADRange As Integer) As Integer

#### @ Argument

**cardNo:** The card number of PCI-9113 card initialized.

**ADRange:** The programmable gain of A/D conversion, the possible values are:

AD\_B\_10\_V: x1

AD\_B\_1\_V: x10

AD\_B\_0\_1\_V: x100

AD\_B\_0\_01\_V: x1000

The information you can get from this argument is gain (x1, x10, x100, or x1000). The actual A/D range depends on the setting of JP1 and JP2 (please refer to section 2.5).

**@ Return Code**

ERR\_NoError

## **6.2.11 \_9113\_AD\_Get\_Status**

**@ Description**

This function is used to get AD FIFO status from the A/D status readback register.

**@ Syntax**

**C/C++ (DOS)**

U16 \_9113\_AD\_Get\_Status (U16 cardNo, U16 \*ADStatus)

**C/C++ (Windows 95)**

U16 W\_9113\_AD\_Get\_Status (U16 cardNo, U16  
\*ADStatus)

**Visual Basic (Windows 95)**

W\_9113\_AD\_Get\_Status (ByVal cardNo As Integer,  
ADStatus As Integer) As Integer

**@ Argument**

**cardNo:** The card number of PCI-9113 card initialized.

**ADStatus:** The status of AD FIFO. The AD FIFO status could be one of the following:

ADSTS\_FF\_EF : FIFO is empty

ADSTS\_FF\_HF : FIFO is half-full

ADSTS\_FF\_FF : FIFO is full, A/D data may have  
been loss

ADSTS\_BUSY : AD is busy, A/D data is written  
into FIFO.

**@ Return Code**

ERR\_NoError

## 6.2.12 \_9113\_AD\_Set\_Mode

### @ Description

This function is used to set A/D trigger mode. Please refer to section 5.1.3 for the detailed description of A/D trigger modes.

### @ Syntax

#### **C/C++ (DOS)**

U16 \_9113\_AD\_Set\_Mode (U16 cardNo, U16 ADMode)

#### **C/C++ (Windows 95)**

U16 W\_9113\_AD\_Set\_Mode (U16 cardNo, U16 ADMode)

#### **Visual Basic (Windows 95)**

W\_9113\_AD\_Set\_Mode (ByVal cardNo As Integer, ByVal ADMode As Integer) As Integer

### @ Argument

**cardNo:** The card number of PCI-9113 card initialized.

**ADMode:** The value of A/D trigger mode.

The mode could be one or a combination of the following modes:

A\_9113\_AD\_FIFO\_ENABLE

A\_9113\_AD\_FIFO\_DISABLE

A\_9113\_AD\_TimerTrig

A\_9113\_AD\_SoftTrig

A\_9113\_AD\_AutoScan

### @ Return Code

ERR\_NoError

## 6.2.13 \_9113\_AD\_Get\_Mode

### @ Description

This function is used to get A/D mode from A/D trigger mode control register. Please refer to section 5.1.3 for the detailed description of A/D trigger modes.

### @ Syntax

**C/C++ (DOS)**

U16 \_9113\_AD\_Get\_Mode (U16 cardNo, U16 \*ADMode)

**C/C++ (Windows 95)**

U16 W\_9113\_AD\_Get\_Mode (U16 cardNo, U16  
\*ADMode)

**Visual Basic (Windows 95)**

W\_9113\_AD\_Get\_Mode (ByVal cardNo As Integer,  
ADMode As Integer) As Integer

**@ Argument**

**cardNo:** The card number of PCI-9113 card initialized.

**ADMode:** The value of A/D trigger mode

The returned value could be one or a combination  
of the following modes:

A\_9113\_AD\_FIFO\_ENABLE

A\_9113\_AD\_FIFO\_DISABLE

A\_9113\_AD\_TimerTrig

A\_9113\_AD\_SoftTrig

A\_9113\_AD\_AutoScan

**@ Return Code**

ERR\_NoError

**6.2.14 \_9113\_INT\_Set\_Reg****@ Description**

This function is used to select the interrupt sources by writing  
data to interrupt control register. Please refer to section 4.8 to  
learn how to set the interrupt control register.

**@ Syntax****C/C++ (DOS)**

U16 \_9113\_INT\_Set\_Reg (U16 cardNo, U16 INTC)

**C/C++ (Windows 95)**

U16 W\_9113\_INT\_Set\_Reg (U16 cardNo, U16 INTC)

**Visual Basic (Windows 95)**

W\_9113\_INT\_Set\_Reg (ByVal cardNo As Integer, ByVal



INTC As Integer) As Integer

**@ Argument**

**cardNo:** The card number of PCI-9113 card initialized.

**INTC:** The value written to the interrupt control register.

**@ Return Code**

ERR\_NoError

### 6.2.15 \_9113\_AD\_Get\_Reg

**@ Description**

This function is used to get the A/D mode setting and interrupt control setting by reading data from the Interrupt control read back register. The settings returned are stored in INTC. Please refer to section 4.8 for the detailed definition of each bit of the returned data.

**@ Syntax**

**C/C++ (DOS)**

U16 \_9113\_INT\_Get\_Reg (U16 cardNo, U16 \*INTC)

**C/C++ (Windows 95)**

U16 W\_9113\_INT\_Get\_Reg (U16 cardNo, U16 \*INTC)

**Visual Basic (Windows 95)**

W\_9113\_INT\_Get\_Reg (ByVal cardNo As Integer, INTC  
As Integer) As Integer

**@ Argument**

**cardNo:** The card number of PCI-9113 card initialized.

**INTC:** The value returned from the interrupt control register.

**@ Return Code**

ERR\_NoError

### 6.2.16 \_9113\_Reset\_FIFO

**@ Description**

The PCI-9113 A/D data are stored in the FIFO after conversion. This function is used to reset A/D FIFO. This function should be called before performing A/D conversion to clear the old data stored in the FIFO.

**@ Syntax**

**C/C++ (DOS)**

U16 \_9113\_Reset\_FIFO (U16 cardNo)

**C/C++ (Windows 95)**

U16 W\_9113\_Reset\_FIFO (U16 cardNo)

**Visual Basic (Windows 95)**

W\_9113\_Reset\_FIFO (ByVal cardNo As Integer) As Integer

**@ Argument**

**cardNo:** The card number of PCI-9113 card initialized.

**@ Return Code**

ERR\_NoError

## **6.2.17 \_9113\_AD\_Soft\_Trigger**

**@ Description**

This function is used to trigger the A/D conversion by software. When this function is called, a trigger pulse will be generated and the converted data will be stored from address Base +0.

**@ Syntax**

**C/C++ (DOS)**

U16 \_9113\_AD\_Soft\_Trigger (U16 cardNo)

**C/C++ (Windows 95)**

U16 W\_9113\_AD\_Soft\_Trigger (U16 cardNo)

**Visual Basic (Windows 95)**

W\_9113\_AD\_Soft\_Trigger (ByVal cardNo As Integer) As Integer

**@ Argument**

**cardNo:** The card number of PCI-9113 card initialized.

**@ Return Code**

ERR\_NoError

## 6.2.18 \_9113\_Set\_8254

**@ Description**

This function is used to write PCI-9113 8254 Programmable Timer.

**@ Syntax**

**C/C++ (DOS)**

U16 \_9113\_Set\_8254 (U16 cardNo, U16 ChannelNo, U8 count)

**C/C++ (Windows 95)**

U16 W\_9113\_Set\_8254 (U16 cardNo, U16 ChannelNo, U8 count)

**Visual Basic (Windows 95)**

W\_9113\_Set\_8254 (ByVal cardNo As Integer, ByVal ChannelNo As Integer, ByVal count As Byte) As Integer

**@ Argument**

**cardNo:** The card number of PCI-9113 card initialized.

**Tmr\_ch:** Port of 8254 Timer, the value is within 0 to 2.

**Count:** The counter value.

**@ Return Code**

ERR\_NoError

### 6.2.19 \_9113\_Get\_8254

#### @ Description

This function is used to read PCI-9113 8254 Programmable Timer. The read value is stored in count.

#### @ Syntax

##### **C/C++ (DOS)**

U16 \_9113\_Get\_8254 (U16 cardNo, U16 ChannelNo, U8 \*count)

##### **C/C++ (Windows 95)**

U16 W\_9113\_Get\_8254 (U16 cardNo, U16 ChannelNo, U8 \*count)

##### **Visual Basic (Windows 95)**

W\_9113\_Get\_8254 (ByVal cardNo As Integer, ByVal ChannelNo As Integer, count As Byte) As Integer

#### @ Argument

**cardNo:** The card number of PCI-9113 card initialized.

**Tmr\_ch:** Port of 8254 Timer, the value is within 0 to 2.

**count:** The value read from 8254 programmable timer, only 8 LSBs are effective

#### @ Return Code

ERR\_NoError

### 6.2.20 \_9113\_AD\_Timer

#### @ Description

This function is used to set the Timer #1 and Timer#2. Timer#1 and Timer#2 are used as frequency dividers for generating constant A/D sampling rate dedicatedly. It is possible to stop the pacer trigger by setting any one of the dividers as 0. Since the A/D conversion rate is limited due to the conversion time of the AD converter, the highest sampling rate of the PCI-9113 can not be exceeded 100 KHz. Thus the multiplication of the dividers must be larger than 20.

**@ Syntax**

**C/C++ (DOS)**

U16 \_9113\_AD\_Timer (U16 cardNo, U16 c1, U16 c2)

**C/C++ (Windows 95)**

U16 W\_9113\_AD\_Timer (U16 cardNo, U16 c1, U16 c2)

**Visual Basic (Windows 95)**

W\_9113\_AD\_Timer (ByVal cardNo As Integer, ByVal c1  
As Integer, ByVal c2 As Integer) As Integer

**@ Argument**

**cardNo:** The card number of PCI-9113 card initialized.

**c1:** frequency divider of timer #1

**c2:** frequency divider of timer #2

**@ Return Code**

ERR\_NoError

## **6.2.21 \_9113\_Counter\_Start**

**@ Description**

The counter #0 of the PCI-9113 Timer/Counter chip can be freely programmed by the users. This function is used to program the counter #0. This counter can be used as frequency generator if internal clock is used. It also can be used as event counter if external clock is used. All the 8254 modes (six operating modes) are available.

**@ Syntax**

**C/C++ (DOS)**

U16 \_9113\_Counter\_Start (U16 cardNo, U16 mode, U16  
c0)

**C/C++ (Windows 95)**

U16 W\_9113\_Counter\_Start (U16 cardNo, U16 mode, U16 c0)

**Visual Basic (Windows 95)**

W\_9113\_Counter\_Start (ByVal cardNo As Integer, ByVal mode As Integer, ByVal c0 As Integer) As Integer

**@ Argument**

**cardNo:** The card number of PCI-9113 card initialized.

**Mode:** the 8254 timer mode, the possible values are:

TIMER\_MODE0, TIMER\_MODE1,  
TIMER\_MODE2, TIMER\_MODE3,  
TIMER\_MODE4, TIMER\_MODE5.

Please refer to Counter/Timer 8254's reference manual for more detailed information of timer mode.

**c0:** counter value of counter#0

**@ Return Code**

ERR\_NoError

**6.2.22 \_9113\_Counter\_Read****@ Description**

This function is used to read the counter value of the Counter#0.

**@ Syntax****C/C++ (DOS)**

U16 \_9113\_Counter\_Read (U16 cardNo, U16 \*c0)

**C/C++ (Windows 95)**

U16 W\_9113\_Counter\_Read (U16 cardNo, U16 \*c0)

**Visual Basic (Windows 95)**

W\_9113\_Counter\_Read (ByVal cardNo As Integer, c0 As Integer) As Integer

**@ Argument**

**cardNo:** The card number of PCI-9113 card initialized.

**c0:** count value of counter#0

**@ Return Code**

ERR\_NoError

### 6.2.23 \_9113\_Counter\_Stop

**@ Description**

This function is used to stop the timer operation. The timer is set as the “One-shot” mode with counter value ‘0’. That is, the clock output signal will be set as high after executing this function.

**@ Syntax**

**C/C++ (DOS)**

U16 \_9113\_Counter\_Stop (U16 cardNo, U16 \*c0)

**C/C++ (Windows 95)**

U16 W\_9113\_Counter\_Stop (U16 cardNo, U16 \*c0)

**Visual Basic (Windows 95)**

U16 W\_9113\_Counter\_Stop (ByVal cardNo As Integer, c0 As Integer) As Integer

**@ Argument**

**cardNo:** The card number of PCI-9113 card initialized.

**c0:** the current counter value of the Counter#0

**@ Return Code**

ERR\_NoError

### 6.2.24 \_9113\_INT\_Source\_Control

**@ Description**

PCI-9113 has a dual-interrupt system, therefore, two interrupt sources can be generated and be checked by the software. This function is used to select and control PCI-9113 interrupt sources by writing data to interrupt control register. Please

refer to section 5.1.4 for detailed description of A/D data transfer modes.

**@ Syntax**

**C/C++ (DOS)**

```
void _9113_INT_Source_Control (U16 cardNo, U16  
int1Ctrl, U16 int2Ctrl)
```

**C/C++ (Windows 95)**

```
void W_9113_INT_Source_Control (U16 cardNo, U16  
int1Ctrl, U16 int2Ctrl)
```

**Visual Basic (Windows 95)**

```
W_9113_INT_Source_Control (ByVal cardNo As Integer,  
ByVal int1Ctrl As Integer, ByVal int2Ctrl As Integer)
```

**@ Argument**

**cardNo:** The card number of PCI-9113 card initialized.

**int1Ctrl:** The value to control INT1, the value can be set and the corresponding definition is the following:

```
int1Ctrl: 0 : INT1 disable  
          1 : INT1 AD end of conversion (EOC)  
            interrupt  
          2 : INT1 FIFO half full
```

**int2Ctrl:** The value to control INT2, the value can be set and the corresponding definition is the following:

```
int2Ctrl: 0 : INT2 disable  
          1 : INT2 pacer timer interrupt  
          2 : INT2 external interrupt source
```

**@ Return Code**

None

## **6.2.25 \_9113\_CLR\_IRQ**

**@ Description**

This function is used to clear interrupt request that is requested by PCI-9113. **If you use interrupt to transfer A/D converted data, you should use this function to clear interrupt**



**request status, otherwise the new coming interrupt will not be generated.**

**@ Syntax**

**C/C++ (DOS)**

void \_9113\_CLR\_IRQ (U16 cardNo)

**C/C++ (Windows 95)**

void W\_9113\_CLR\_IRQ (U16 cardNo)

**Visual (Windows 95)**

W\_9113\_CLR\_IRQ (ByVal cardNo As Integer)

**@ Argument**

None

**@ Return Code**

None

## **6.2.26 \_9113\_Get\_IRQ\_Channel**

**@ Description**

This function is used to get the IRQ level of the PCI-9113 card currently used.

**@ Syntax**

**C/C++ (DOS)**

void \_9113\_Get\_IRQ\_Channel (U16 cardNo, U16 \*irq\_no)

**C/C++ (Windows 95)**

void W\_9113\_Get\_IRQ\_Channel (U16 cardNo, U16  
\*irq\_no)

**Visual Basic (Windows 95)**

W\_9113\_Get\_IRQ\_Channel (ByVal cardNo As Integer,  
irq\_no As Integer)

**@ Argument**

**cardNo:** The card number of PCI-9113 card initialized.

**Irq\_no:** The IRQ level used to transfer A/D data for this card.

**@ Return Code**

None

### 6.2.27 \_9113\_Get\_IRQ\_Status

#### @ Description

This function is used to get the status of the two IRQs (INT1 and INT2) in PCI-9113 card.

#### @ Syntax

##### C/C++ (DOS)

```
void _9113_Get_IRQ_Status (U16 cardNo, U16 *ch1, U16  
*ch2)
```

##### C/C++ (Windows 95)

```
void W_9113_Get_IRQ_Status (U16 cardNo, U16 *ch1,  
U16 *ch2)
```

##### Visual Basic (Windows 95)

```
W_9113_Get_IRQ_Status (ByVal cardNo As Integer, ch1  
As Integer, ch2 As Integer)
```

#### @ Argument

**cardNo:** the card number of PCI-9113 card initialized.

**ch1:** the IRQ status of INT1

**ch2:** the IRQ status of INT2

#### @ Return Code

None

### 6.2.28 \_9113\_AD\_FFHF\_Polling

#### @ Description

This function is used to perform the powerful AD data transfer by applying half-full polling mode. This method checks the FIFO half full signal every time calling this function. If the FIFO is not half-full, the software does not read data. When the FIFO is full, the AD FIFO is overrun. When the FIFO is half-full but not full, software reads the A/D data, which is stored in FIFO, in size of one "block" (512 words). The FIFO

half-full polling method is the most powerful A/D data transfer mode. Please refer to section 5.1.4 for the detailed description of half-full polling mode.

#### @ Syntax

##### **C/C++ (DOS)**

U16 \_9113\_AD\_FFHF\_Polling (U16 cardNo, U16 far  
\*ad\_buf)

##### **C/C++ (Windows 95)**

U16 W\_9113\_AD\_FFHF\_Polling (U16 cardNo, U16  
\*ad\_buf)

##### **Visual Basic (Windows 95)**

W\_9113\_AD\_FFHF\_Polling (ByVal cardNo As Integer,  
ad\_buf As Integer) As Integer

#### @ Argument

**cardNo:** The card number of PCI-9113 card initialized.

**ad\_buf:** The buffer stores the A/D converted value. The size of ad\_buf can not be smaller than 512 words. The data format can be referred to section 5.1.5 for the details.

#### @ Return Code

ERR\_NoError

ERR\_FIFO\_Half\_NotReady

### 6.2.29 \_9113\_AD\_FFHF\_Polling\_MUX

#### @ Description

This function is used to perform powerful AD data transfer by applying half-full polling mode. This method checks the FIFO half full signal every time calling this function. If the FIFO is not half-full, the software does not read data. When the FIFO is full, the AD FIFO is overrun. When the FIFO is half-full and not full, software reads the A/D data, which is stored in FIFO, in size of one "block" (512 words). The difference between this function and 9113\_AD\_FFHF\_Polling is that the former reads data from the 16-bit register and the latter reads data from 32-

bit data register. Please refer to section 5.1.4 for the detailed description of half-full polling mode.

**@ Syntax**

**C/C++ (DOS)**

U16 \_9113\_AD\_FFHF\_Polling\_MUX (U16 cardNo, U32 far \*ad\_buf)

**C/C++ (Windows 95)**

U16 W\_9113\_AD\_FFHF\_Polling\_MUX (U16 cardNo, U32 \*ad\_buf)

**Visual Basic (Windows 95)**

U16 W\_9113\_AD\_FFHF\_Polling\_MUX (ByVal cardNo As Integer, ad\_buf As Long) As Integer

**@ Argument**

**cardNo:** The card number of PCI-9113 card initialized.

**ad\_buf:** The 32bits A/D converted value. The data format can be referred to section 5.1.5 for details.

**@ Return Code**

ERR\_NoError

ERR\_FIFO\_Half\_NotReady

## 6.2.30 \_9113\_AD\_Aquire

**@ Description**

This function is used to poll the A/D converted data for PCI-9113 by software trigger. It reads the A/D data when the data is ready.

**@ Syntax**

**C/C++ (DOS)**

U16 \_9113\_AD\_Aquire (U16 cardNo, U16 far \*ad\_data)

**C/C++ (Windows 95)**

U16 W\_9113\_AD\_Aquire (U16 cardNo, U16 \*ad\_data)

**Visual Basic (Windows 95)**

W\_9113\_AD\_Aquire (ByVal cardNo As Integer, ad\_data

As Integer) As Integer

**@ Argument**

**cardNo:** The card number of PCI-9113 card initialized.

**ad\_data:** The 16-bit A/D converted value. The bit 0 of ADData is the LSB of A/D converted data and the bit 11 of ADData is the MSB of A/D converted data. Please refer to section 5.1.5 for the relationship between the voltage and the value.

**@ Return Code**

ERR\_NoError

ERR\_AD\_AquireTimeOut

## **6.2.31 \_9113\_AD\_Aquire\_MUX**

**@ Description**

This function is used to poll the A/D conversion data for PCI-9113. It reads the A/D data when the data is ready.

**@ Syntax**

**C/C++ (DOS)**

U16 \_9113\_AD\_FFHF\_Polling\_MUX (U16 cardNo, U16 far \*ad\_data)

**C/C++ (Windows 95)**

U16 W\_9113\_AD\_FFHF\_Polling\_MUX (U16 cardNo, U16 \*ad\_data)

**Visual Basic (Windows 95)**

W\_9113\_AD\_FFHF\_Polling\_MUX (ByVal cardNo As Integer, ad\_data As Integer) As Integer

**@ Argument**

**cardNo:** The card number of PCI-9113 card initialized.

**ad\_data:** The 32-bit A/D converted value. The resolution of A/D conversion data is 12 bits. The unsigned integer data format of ADData is as follows:  
Every 32-bit unsigned integer data:  
bit 0...11: A/D converted data

bit 16, 17, ..., 20 : converted channel no.  
Please refer to section 5.1.5 for the relationship  
between the voltage and the value.

**@ Return Code**

ERR\_NoError  
ERR\_FIFO\_Half\_NotReady

## **6.2.32 \_9113\_AD\_INT\_Start**

**@ Description**

This function is used to initial and startup the AD EOC (end-of-conversion) interrupt. This function could perform A/D conversion N times with interrupt data transfer by using pacer trigger. It takes place in the background and will not stop until the N-th conversion has been completed or your program execute \_9113\_AD\_INT\_Stop() function to stop the process. After executing this function, it is necessary to check the status of the operation by using the function \_9113\_AD\_INT\_Status(). The function can perform on single A/D channel (autoscan is disabled) or multiple A/D channels (autoscan is enabled) with a fixed analog input range.

---

**Note:** The interrupt mode provided in this function is internal timer source, therefore you must specify c1 & c2 as calling this function. In addition, this function in MS-DOS Borland C++ library supports just one PCI-9113 card and provides only one ISR (interrupt service routine) for processing the interrupt events. If multi-9113 cards and multi-isr is necessary, users can modify this library for your own purpose.

---

**@ Syntax**

**C/C++ (DOS)**

U16 \_9113\_AD\_INT\_Start (U16 cardNo, U16 auto\_scan,  
U16 ad\_ch\_no, U16 ad\_gain, U16 count, U32  
\*ad\_buffer, U16 c1, U16 c2)

### C/C++ (Windows 95)

U16 W\_9113\_AD\_INT\_Start (U16 cardNo, U16 auto\_scan,  
U16 ad\_ch\_no, U16 ad\_gain, U16 count, U32  
\*ad\_buffer, U16 c1, U16 c2)

### Visual Basic (Windows 95)

W\_9113\_AD\_INT\_Start (ByVal cardNo As Integer, ByVal  
auto\_scan As Integer, ByVal ad\_ch\_no As Integer,  
ByVal ad\_gain As Integer, ByVal count As Integer,  
ad\_buffer As Long, ByVal c1 As Integer, ByVal c2 As  
Integer) As Integer

### @ Argument

**cardNo:** The card number of PCI-9113 card initialized.

**auto\_scan:** 0: autoscan is disabled.

1: autoscan is enabled.

**ad\_ch\_no:** A/D channel number.

If the auto\_scan is set as enabled, the selection  
sequence of A/D channel is: 0, 1, 2, 3, ...,

[ad\_ch\_no], 0, 1, 2, 3, [ad\_ch\_no], ...

If the auto\_scan is set as disabled, only the data  
input from [ad\_ch\_no] is converted.

**ad\_gain:** A/D analog input range, the possible values are:

AD\_B\_10\_V, AD\_B\_1\_V, AD\_B\_0\_1\_V,

AD\_B\_0\_01\_V, AD\_B\_5\_V, AD\_B\_0\_5\_V,

AD\_B\_0\_05\_V, AD\_B\_0\_005\_V, AD\_U\_10\_V,

AD\_U\_1\_V, AD\_U\_0\_1\_V, AD\_U\_0\_01\_V.

**count:** The number of A/D conversion

**ad\_buffer:** The start address of the memory buffer to store  
the A/D data. The buffer size must large than the  
number of A/D conversion. The unsigned integer  
data format in ad\_buffer is as follows:

Every 32-bit unsigned integer data:

bit 0...11: A/D converted data

bit 16, 17, ..., 20 : converted channel no.

Please refer to section 5.1.5 for the relationship  
between the voltage and the value.

**c1:** the frequency divider of Timer#1

**c2:** the frequency divider of Timer#2

**@ Return Code**

ERR\_InvalidADChannel  
ERR\_AD\_InvalidGain  
ERR\_InvalidTimerValue  
ERR\_NoError

**6.2.33    \_9113\_AD\_FFHF\_INT\_Start**

**@ Description**

This function is used to initial and start up the interrupt transfer by using AD FIFO Half-Full Interrupt Transfer Mode. This function could perform A/D conversion N times with interrupt data transfer by using pacer trigger. It takes place in the background and will not stop until the N-th conversion has been completed or your program execute \_9113\_AD\_INT\_Stop() function to stop the process. After executing this function, it is necessary to check the status of the operation by using the function \_9113\_AD\_FFHF\_INT\_Status(). The function can perform on single A/D channel (autoscan is disabled) or multiple A/D channels (autoscan is enabled) with fixed analog input range.

---

**Note:** The interrupt mode provided in this function is internal timer source, therefore you must specify c1 & c2 as calling this function. In addition, this function in MS-DOS Borland C++ library supports just one PCI-9113 card and provides only one ISR (interrupt service routine) for processing the interrupt events. If multi-9113 cards and multi-isr is necessary, users can modify this library for your own purpose.

---

**@ Syntax**

**C/C++ (DOS)**

U16 \_9113\_AD\_FFHF\_INT\_Start (U16 cardNo, U16  
    auto\_scan, U16 ad\_ch\_no, U16 ad\_gain, U16 blockNo,  
    U32 \*ad\_buffer, U16 c1, U16 c2)



### C/C++ (Windows 95)

U16 W\_9113\_AD\_FFHF\_INT\_Start (U16 cardNo, U16 auto\_scan, U16 ad\_ch\_no, U16 ad\_gain, U16 blockNo, U32 \*ad\_buffer, U16 c1, U16 c2)

### Visual Basic (Windows 95)

W\_9113\_AD\_FFHF\_INT\_Start (ByVal cardNo As Integer, ByVal auto\_scan As Integer, ByVal ad\_ch\_no As Integer, ByVal ad\_gain As Integer, ByVal blockNo As Integer, ad\_buffer As Long, ByVal c1 As Integer, ByVal c2 As Integer) As Integer

### @ Argument

**cardNo:** The card number of PCI-9113 card initialized.

**auto\_scan:** 0: autoscan is disabled.

1: autoscan is enabled.

**ad\_ch\_no:** A/D channel number.

If the auto\_scan is set as enable, the selection sequence of A/D channel is: 0, 1, 2, 3, ...,

[ad\_ch\_no], 0, 1, 2, 3, [ad\_ch\_no], ...

If the auto\_scan is set as disable, only the data input from [ad\_ch\_no] is converted.

**ad\_gain:** A/D analog input range, the possible values are:

AD\_B\_10\_V, AD\_B\_1\_V, AD\_B\_0\_1\_V,

AD\_B\_0\_01\_V, AD\_B\_5\_V, AD\_B\_0\_5\_V,

AD\_B\_0\_05\_V, AD\_B\_0\_005\_V, AD\_U\_10\_V,

AD\_U\_1\_V, AD\_U\_0\_1\_V, AD\_U\_0\_01\_V.

**blockNo:** The number of blocks for performing A/D conversion, one block of A/D conversion is 512 words.

**ad\_buffer:** The start address of the memory buffer to store the AD data. The buffer size must large than the number of AD conversion. The unsigned integer data format in ad\_buffer is as follows:

Every 32-bit unsigned integer data:

bit 0...11: A/D converted data

bit 16, 17, ..., 20 : converted channel no.

Please refer to section 5.1.5 for the relationship between the voltage and the value.

**c1:** the frequency divider of Timer#1  
**c2:** the frequency divider of Timer#2

**@ Return Code**

ERR\_InvalidADChannel  
ERR\_AD\_InvalidGain  
ERR\_InvalidTimerValue  
ERR\_NoError

### 6.2.34 \_9113\_AD\_INT\_Status

**@ Description**

This function is used to check the status of interrupt operation. The \_9113\_AD\_INT\_Start() is executed on background, therefore you can issue this function to check the status of interrupt operation.

**@ Syntax**

**C/C++ (DOS)**

U16 \_9113\_AD\_INT\_Status (U16 cardNo, U16 \*status,  
U16 \*count)

**C/C++ (Windows 95)**

U16 W\_9113\_AD\_INT\_Status (U16 cardNo, U16 \*status,  
U16 \*count)

**Visual Basic (Windows 95)**

W\_9113\_AD\_INT\_Status (ByVal cardNo As Integer, status  
As Integer, count As Integer) As Integer

**@ Argument**

**cardNo:** The card number of PCI-9113 card initialized.

**status:** The status of the INT data transfer.

**count:** The A/D conversion count number performed currently.

**@ Return Code**

ERR\_NoError

### 6.2.35 \_9113\_AD\_FFHF\_INT\_Status

#### @ Description

This function is used to check the status of interrupt operation by using AD FIFO Half Full Interrupt Transfer Mode. The \_9113\_AD\_FFHF\_INT\_Start() is executed on background, therefore you can issue this function to check the status of interrupt operation.

#### @ Syntax

##### C/C++ (DOS)

U16 \_9113\_AD\_FFHF\_INT\_Status (U16 cardNo, U16 \*status, U16 \*blockNo)

##### C/C++ (Windows 95)

U16 W\_9113\_AD\_FFHF\_INT\_Status (U16 cardNo, U16 \*status, U16 \*blockNo)

##### Visual Basic (Windows 95)

W\_9113\_AD\_FFHF\_INT\_Status (ByVal cardNo As Integer, status As Integer, blockNo As Integer) As Integer

#### @ Argument

**cardNo:** The card number of PCI-9113 card initialized.

**status:** The status of the INT data transfer.

**blockno:** The A/D conversion block number performed currently.

#### @ Return Code

ERR\_NoError

### 6.2.36 \_9113\_AD\_FFHF\_INT\_Restart

#### @ Description

After calling \_9113\_AD\_FFHF\_INT\_Start(), the A/D conversion and transfer won't stop until the N blocks of the A/D data is acquired, calling this function can restart the FIFO half full interrupt transfer without re-initial all the relative

registers. However, if the interrupt operation was stopped by calling `_9113_AD_FFHF_INT_Stop()`, the program should use `_9113_AD_FFHF_INT_Start()` to restart the interrupt transfer function.

**@ Syntax**

**C/C++ (DOS)**

`U16 _9113_AD_FFHF_INT_Restart (U16 cardNo)`

**C/C++ (Windows 95)**

`U16 W_9113_AD_FFHF_INT_Restart (U16 cardNo)`

**Visual Basic (Windows 95)**

`W_9113_AD_FFHF_INT_Restart (ByVal cardNo As Integer) As Integer`

**@ Argument**

**cardNo:** The card number of PCI-9113 card initialized.

**@ Return Code**

`ERR_NoError`

### 6.2.37 \_9113\_AD\_INT\_Stop

#### @ Description

This function is used to stop the interrupt data transfer function. After executing this function, the internal A/D trigger is disabled and the A/D timer is stopped. This function returns the number of data has been transferred, no matter whether the A/D interrupt data transfer is stopped by this function.

#### @ Syntax

##### **C/C++ (DOS)**

U16 \_9113\_AD\_INT\_Stop (U16 cardNo, U16 \*count)

##### **C/C++ (Windows 95)**

U16 W\_9113\_AD\_INT\_Stop (U16 cardNo, U16 \*count)

##### **Visual Basic (Windows 95)**

W\_9113\_AD\_INT\_Stop (ByVal cardNo As Integer, count  
As Integer) As Integer

#### @ Argument

**CardNo:** The card number of PCI-9113 card initialized.

**count:** The number of A/D data which has been transferred.

#### @ Return Code

ERR\_AD\_INTNotSet

ERR\_NoError



## Calibration & Utilities

In data acquisition process, how to calibrate your measurement devices to maintain its accuracy is very important. Users can calibrate the analog input and analog output channels under the users' operating environment for optimizing the accuracy. This chapter will guide you to calibrate your PCI-9113 to an accuracy condition.

---

### 7.1 What do you need

Before calibrating your PCI-9113 card, you should prepare some equipment's for the calibration:

- Calibration program : Once the program is executed, it will guide you to do the calibration. This program is included in the delivered package.
- A 5 1/2 digit multimeter (6 1/2 is recommended)
- A voltage calibrator or a very stable and noise free DC voltage generator.

---

## 7.2 VR Assignment

There are three variable resistors (VR) on the PCI-9113 board to allow you making accurate adjustment on A/D channels. The function of each VR is specified as Table 7.1.

VR1	A/D uni-polar offset adjustment
VR2	A/D full scale adjustment
VR3	A/D bi-polar offset adjustment

**Table 7.1 Function of VRs**

---

## 7.3 A/D Adjustment

### 7.3.1 Uni-polar input

1. Set the analog gain = 1 and channel number #0 by software.
2. Set JP1 as uni-polar A/D input. Set JP2 to 10V full range.
3. Short the A/D channel 0 (pin 1 of CN1) to ground (GND).  
Adjust **VR1** to obtain reading between 0~1.
4. Applied a +10V reference input signal to A/D channel 1 (pin 20 of CN1), and trim the **VR2** to obtain reading between 4094~4095.

### 7.3.2 Bi-polar input

1. Set the analog gain = 1 and channel number #0 by software.
2. Set JP1 as bi-polar A/D input. Set JP2 to 20V full range.
3. Short the A/D channel 0 (pin 1 of CN1) to ground (GND).  
Adjust **VR3** to obtain reading between 2047~2048.
4. Applied a +10V reference input signal to A/D channel 1 (pin 20 of CN1), and trim the **VR2** to obtain reading between 4094~4095.



---

## 7.4 Software A/D Offset Calibration

For more accuracy calibrate the input offset signal, using software to calibrate the offset of the analog input signal is a good approach. Another benefit is this method can calibrate offset online and thus eliminate any temperature drift. For example, user can short the resistor RB32 to ground. Measuring the digital value of channel #31 can obtain the offset voltage of the AD channels. If the digital offset value is  $V_{off}$ , user can modify any AD data by subtracting  $V_{off}$  from the AD data to obtain the offset calibrated value. Note that the  $V_{off}$  may be different for each gain level,. Users should calibrate the offset value for every gain value.



## Software Utility

This software CD provides two utility programs. They are 9113util.exe which provides three functions, System Configuration, Calibration, and Functional Testing, and I\_eeeprom which is used to enable or disable interrupt of PCI-9113 board. The utility programs are described in the following sections.

---

### 8.1 9113util

There are three functions provided by 9113util. They are System Configuration, Calibration, and Functional Testing. This utility software is designed as menu-driven based windowing style. Not only the text messages are shown for operating guidance, but also has the graphic to indicate you how to set right hardware configuration.

#### 8.1.1 Running 9113util.exe

After finishing the DOS installation, you can execute the utility by typing as follows :

```
C> cd \ADLINK\DOS\9113\Util  
C> 9113UTIL
```

The following diagram will be displayed on you screen. The message at the bottom of each window guides you how to select item, go to the next step and change the default settings.

```
***** PCI-9113 Utility  Rev. 1.0 *****  
Copyright © 1995-1996, ADLink Technology Inc.  All rights reserved.
```

```
<F1> : Configuration.  
<F2> : Calibration.  
<F3> : Function testing.  
<Esc>: Quit.
```

```
>>> Select function key F1 ~ F3, or press <Esc> to quit. <<<
```

### 8.1.2 System Configuration

This function guides you to configure the PCI-9113 card, and set the right hardware configuration. The configuration window shows the setting items that you have to set before using the PCI-9113 card.

The following diagram will be displayed on the screen as you choose the Configuration function from main menu.

\*\*\*\*\* Calibration of PCI9113 \*\*\*\*\*

<1> Card Type	PCI9113
<2> AD Polarity setting	Bipolar
<3> AD Input Range	Gain=1 Bipolar(-10V~10V)

>>> <Up/Down>: Select Item, <PgUp/PgDn>: Change Setting <<<

### 8.1.3 Calibration

This function guides you to calibrate the PCI-9113. The calibration program serves as a useful test of the PCI-9113's A/D and D/A functions and can aid in troubleshooting if problems arise.

---

**Note:** For an environment with frequently large changes of temperature and vibration, a 3 months re-calibration interval is recommended. For laboratory conditions, 6 months to 1 year is acceptable

---

When you choose the calibration function from the main menu list, a calibration items menu is displayed on the screen. After you select one of the calibration items from the calibration items menu, a calibration window shows. The upper window shows the detailed procedures which have to be followed when you proceed the calibration. The instructions will guide you to calibrate each item step by step. The bottom window shows the layout of PCI-9113. In addition, the proper Variable Resister (VR) will blink to indicate the related VR which needs to be adjusted for the current calibration step.

\*\*\*\*\* PCI-9113 Calibration \*\*\*\*\*

<1> A/D Bipolar (Gain = 1, -10V ~ 10V) adjusting  
<2> A/D Unipolar (Gain = 1, 0V ~ 10V) adjusting  
<Esc> Quit

Select 1 to 2 or <Esc> to quit calibration.

If you select 1, the following figure displays on the screen:

**A/D Bipolar (Gain = 1, -10V ~ 10V) adjusting**

Step 1: Set JP1 as bi-polar A/D input. Set JP2 to 20V full range.  
Step 2: Connect A/D channel 0 ( pin 1 of CN1) to GND(pin10 of CN1).  
Step 3: Trim VR3 until the AD value is jumping from 2047 to 2048  
Step 4: Input 10V to A/D channel 0 (pin1 of CN1)  
Step 5: Trim VR2 until the AD value is jumping from 4094 to 4095

**CH0 = 4095**

1 2 3  
| | |  
JP1 ☐  
JP2 ☐

**<F10> Completed AD calibration, otherwise repeat Step 2 to 5.**

**A/D Bipolar (Gain = 1, -10V ~ 10V) adjusting**

#### 8.1.4 Functional Testing

This function is used to test the functions of PCI-9113. It includes Digital I/O testing, D/A testing, A/D polling testing, A/D Interrupt Testing, and A/D FIFO Half-Full Interrupt testing.

When you choose one of the testing functions from the functions menu, a diagram is displayed on the screen. The figures below are the function testing menu window and A/D with polling Testing window.

```
***** PCI-9113 Function Testing *****
```

```
<1> : A/D with Polling Test  
<2> : A/D with Interrupt Test  
<3> : A/D with FIFO Half-Full Interrupt  
<Esc>: Quit
```

Select 1 to 3 or <Esc> to quit function testing

#### Function Testing Menu Window

\*\*\*\*\* PCI-9113 Utility Rev. 1.0 \*\*\*\*\*

Copyright (c) 1995-1997, ADLink Technology Inc. All rights reserved.

CH#	AD Data	Volt	CH#	AD Data	Volt
0	1372	-3.2991	16	1368	-3.3187
1	1373	-3.2943	17	1373	-3.2943
2	1369	-3.3138	18	1373	-3.2943
3	1373	-3.2943	19	1382	-3.2503
4	1363	-3.3431	20	1377	-3.2747
5	1362	-3.3480	21	1382	-3.2503
6	1358	-3.3675	22	1378	-3.2698
7	1332	-3.4945	23	1376	-3.2796
8	1341	-3.4505	24	1386	-3.2308
9	1344	-3.4359	25	1390	-3.2112
10	1344	-3.4359	26	1385	-3.2357
11	1351	-3.4017	27	1389	-3.2161
12	1345	-3.4310	28	1382	-3.2503
13	1352	-3.3968	29	1385	-3.2357
14	1353	-3.3919	30	1382	-3.2503
15	1362	-3.3480	31	1385	-3.2357

>>> Press any key to stop AD conversion <<<

### A/D with FIFO Half-Full Interrupt test window

## 8.2 I\_EEPROM

This file is used to enable or disable the interrupt of PCI-9113 board. This software is a text-driven program. Because the default interrupt on PCI-9113 board is "on", users who don't want to use interrupt function can use this utility to turn off the interrupt of their PCI-9113 board.

### 8.2.1 Running I\_eeeprom.exe

After finishing the DOS installation, you can execute the utility by typing as follows :

```
C> cd \ADLINK\DOS\9113\UTIL
```

```
C> I_eeeprom
```

At first, this program prompts you to input the card type—9113. After specifying the card type, this program shows the instructions to guide you to enable or disable the interrupt of your PCI-9113



board.



# Product Warranty/Service

Seller warrants that equipment furnished will be free from defects in material and workmanship for a period of one year from the confirmed date of purchase of the original buyer and that upon written notice of any such defect, Seller will, at its option, repair or replace the defective item under the terms of this warranty, subject to the provisions and specific exclusions listed herein.

This warranty shall not apply to equipment that has been previously repaired or altered outside our plant in any way as to, in the judgment of the manufacturer, affect its reliability. Nor will it apply if the equipment has been used in a manner exceeding its specifications or if the serial number has been removed.

Seller does not assume any liability for consequential damages as a result from our products uses, and in any event our liability shall not exceed the original selling price of the equipment.

The equipment warranty shall constitute the sole and exclusive remedy of any Buyer of Seller equipment and the sole and exclusive liability of the Seller, its successors or assigns, in connection with equipment purchased and in lieu of all other warranties expressed implied or statutory, including, but not limited to, any implied warranty of merchant ability or fitness and all other obligations or liabilities of seller, its successors or assigns.

The equipment must be returned postage-prepaid. Package it securely and insure it. You will be charged for parts and labor if you lack proof of date of purchase, or if the warranty period is expired.